

Computer Networks

Chapter 4 Network Layer

Adapted from Book Slides: “Computer Networking: A Top-Down Approach”

Some slides copyright J.F Kurose and K.W. Ross, All Rights Reserved

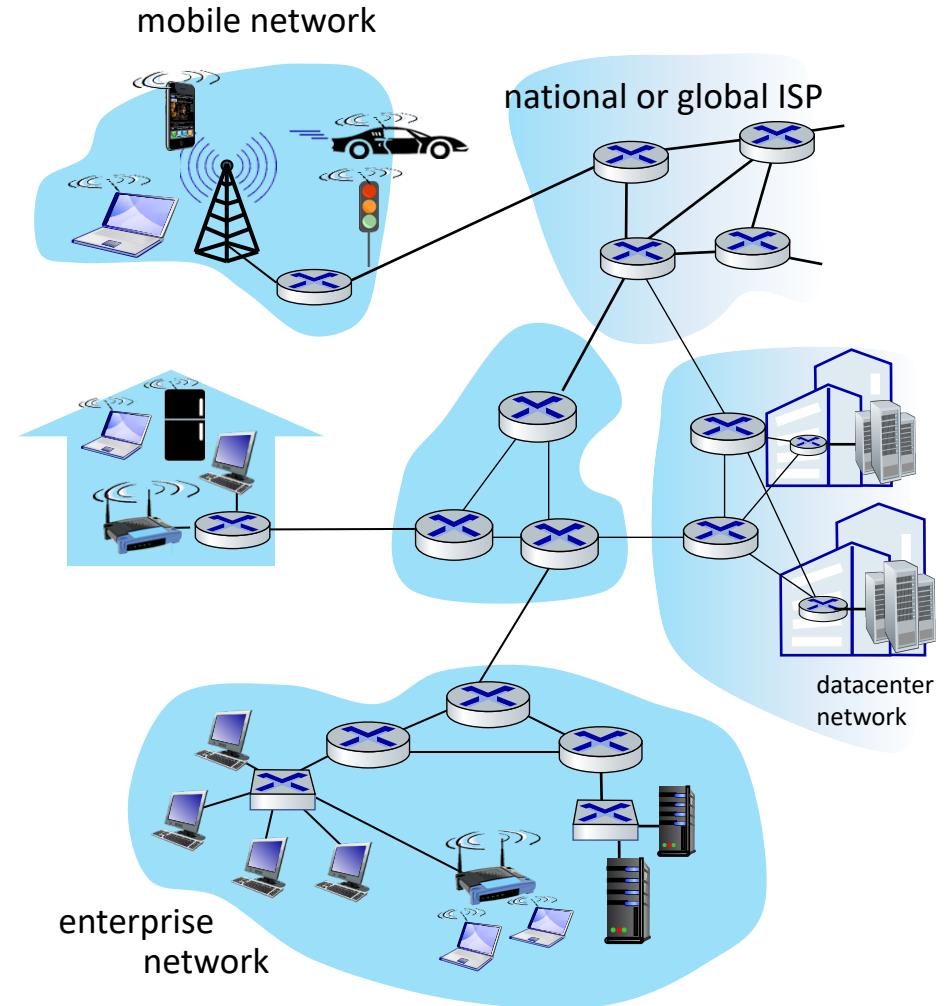
Chapter 4: our goals

- understand principles behind network layer services, focusing on data plane:
 - network layer service models
 - forwarding versus routing
 - how a router works
 - addressing
 - generalized forwarding
 - Internet architecture

Chapter 4: roadmap

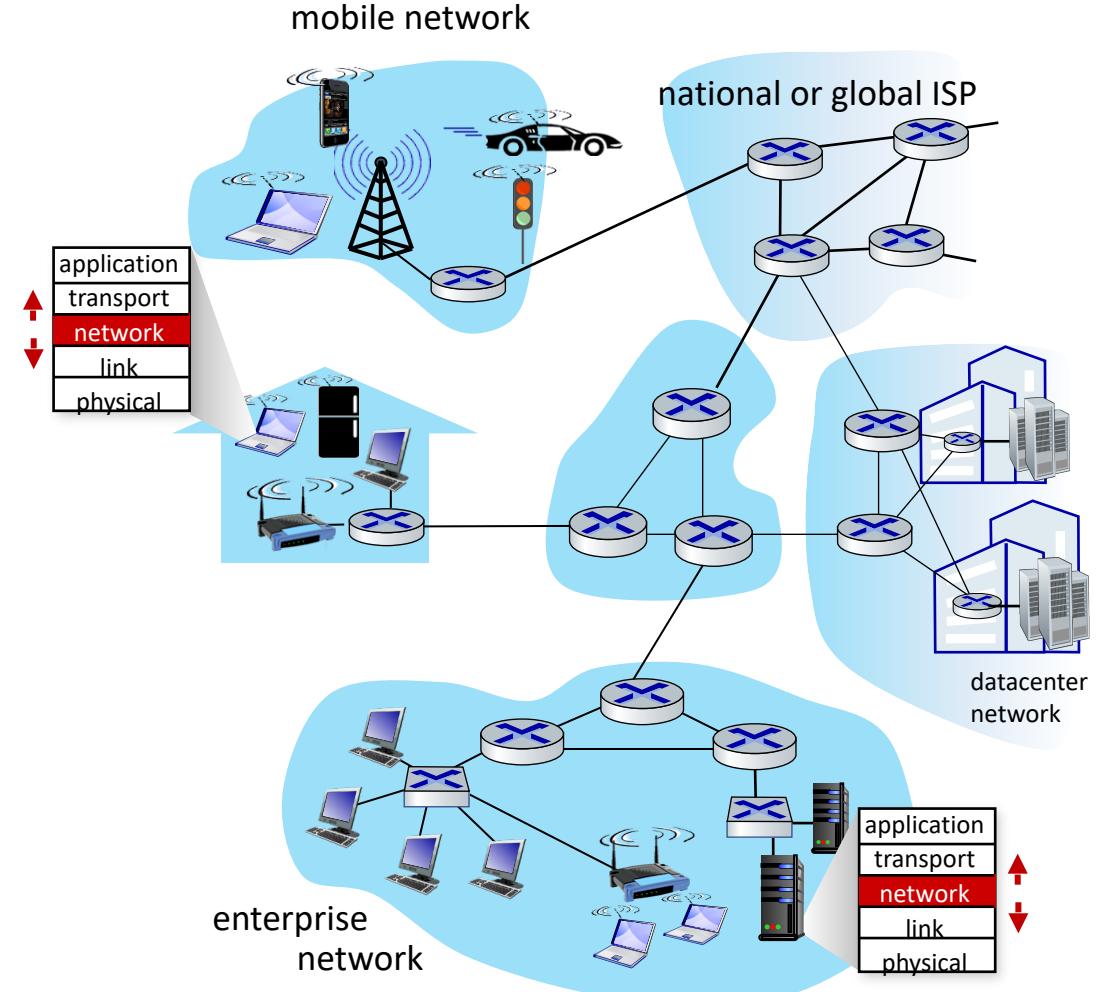
- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

Network-layer services and protocols



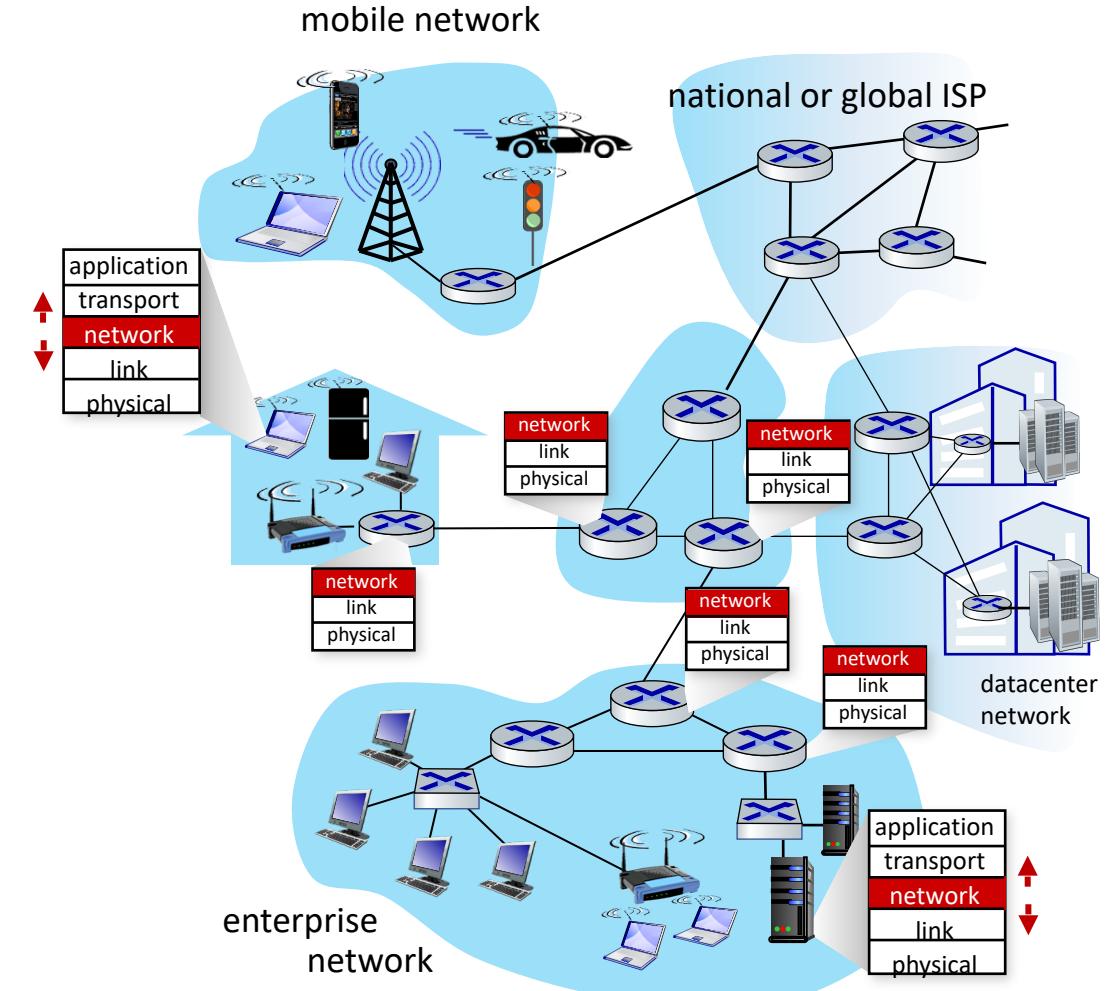
Network-layer services and protocols

- transport segment from sending to receiving host
 - **sender:** encapsulates segments into datagrams, passes to link layer
 - **receiver:** delivers segments to transport layer protocol



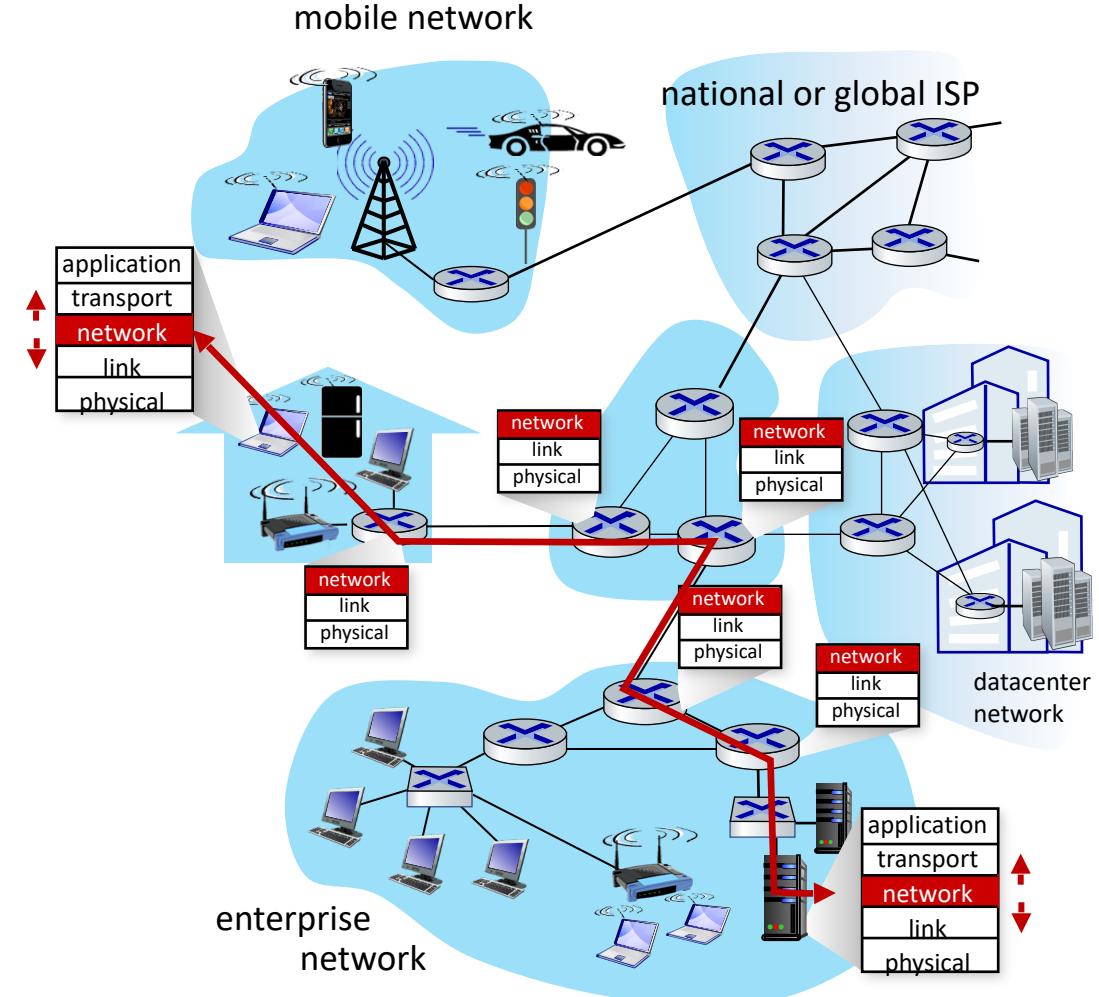
Network-layer services and protocols

- transport segment from sending to receiving host
 - **sender:** encapsulates segments into datagrams, passes to link layer
 - **receiver:** delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers



Network-layer services and protocols

- transport segment from sending to receiving host
 - **sender:** encapsulates segments into datagrams, passes to link layer
 - **receiver:** delivers segments to transport layer protocol
- network layer protocols in *every Internet device*: hosts, routers
- **routers:**
 - examines header fields in all IP datagrams passing through it
 - moves datagrams from input ports to output ports to transfer datagrams along end-end path



Two key network-layer functions

network-layer functions:

- *forwarding*: move packets from a router's input link to appropriate router output link

Two key network-layer functions

network-layer functions:

- *forwarding*: move packets from a router's input link to appropriate router output link
- *routing*: determine route taken by packets from source to destination
 - *routing algorithms*

Two key network-layer functions

network-layer functions:

- *forwarding*: move packets from a router's input link to appropriate router output link
- *routing*: determine route taken by packets from source to destination
 - *routing algorithms*

analogy: taking a trip

- *forwarding*: process of getting through single interchange



forwarding

Two key network-layer functions

network-layer functions:

- **forwarding**: move packets from a router's input link to appropriate router output link
- **routing**: determine route taken by packets from source to destination
 - *routing algorithms*

analogy: taking a trip

- **forwarding**: process of getting through single interchange
- **routing**: process of planning trip from source to destination



forwarding

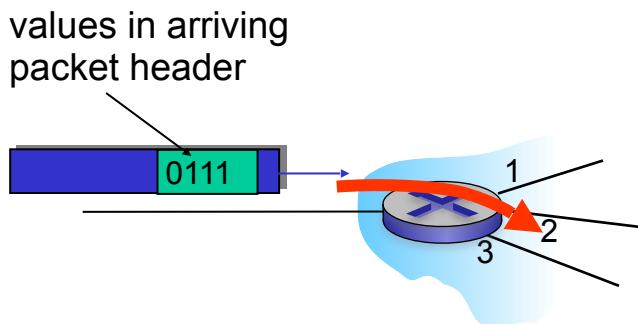


routing

Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



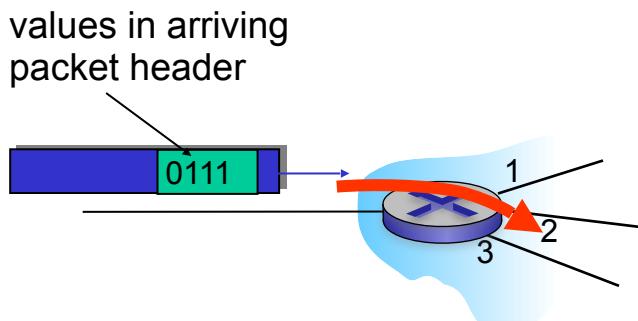
Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

Control plane

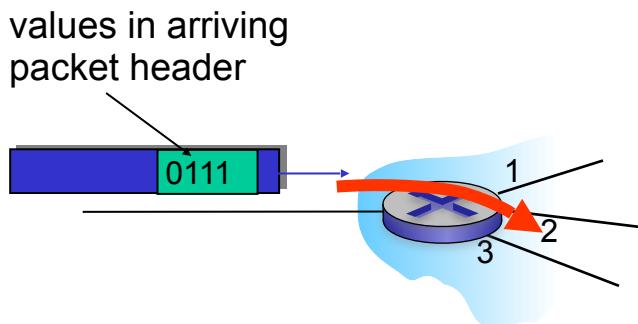
- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host



Network layer: data plane, control plane

Data plane:

- *local*, per-router function
- determines how datagram arriving on router input port is forwarded to router output port

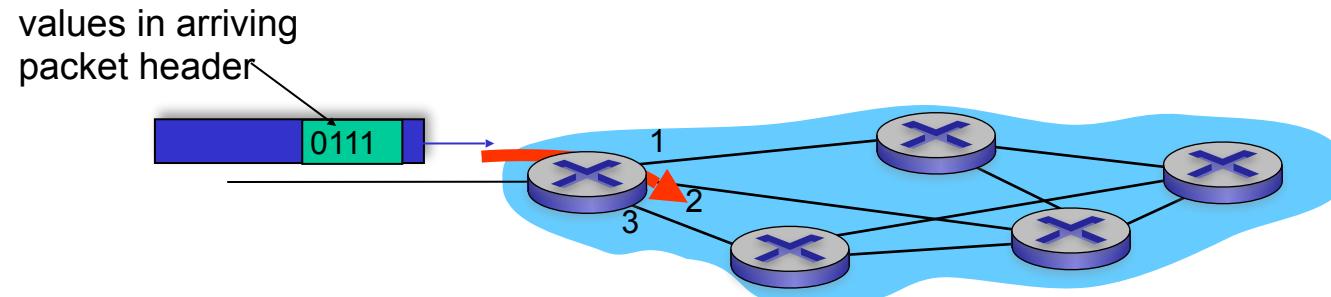


Control plane

- *network-wide* logic
- determines how datagram is routed among routers along end-end path from source host to destination host
- two control-plane approaches:
 - *traditional routing algorithms*: implemented in routers
 - *software-defined networking (SDN)*: implemented in (remote) servers

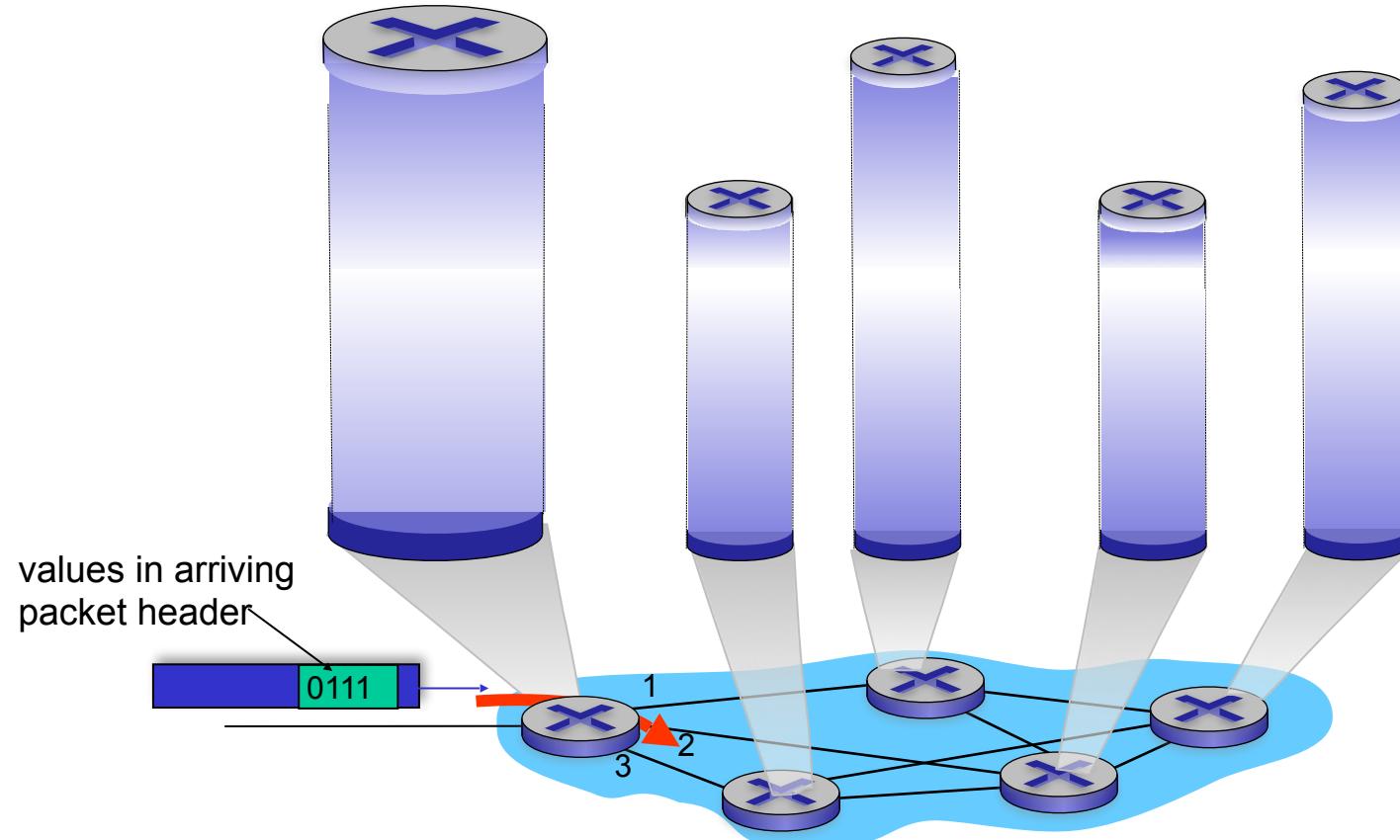
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



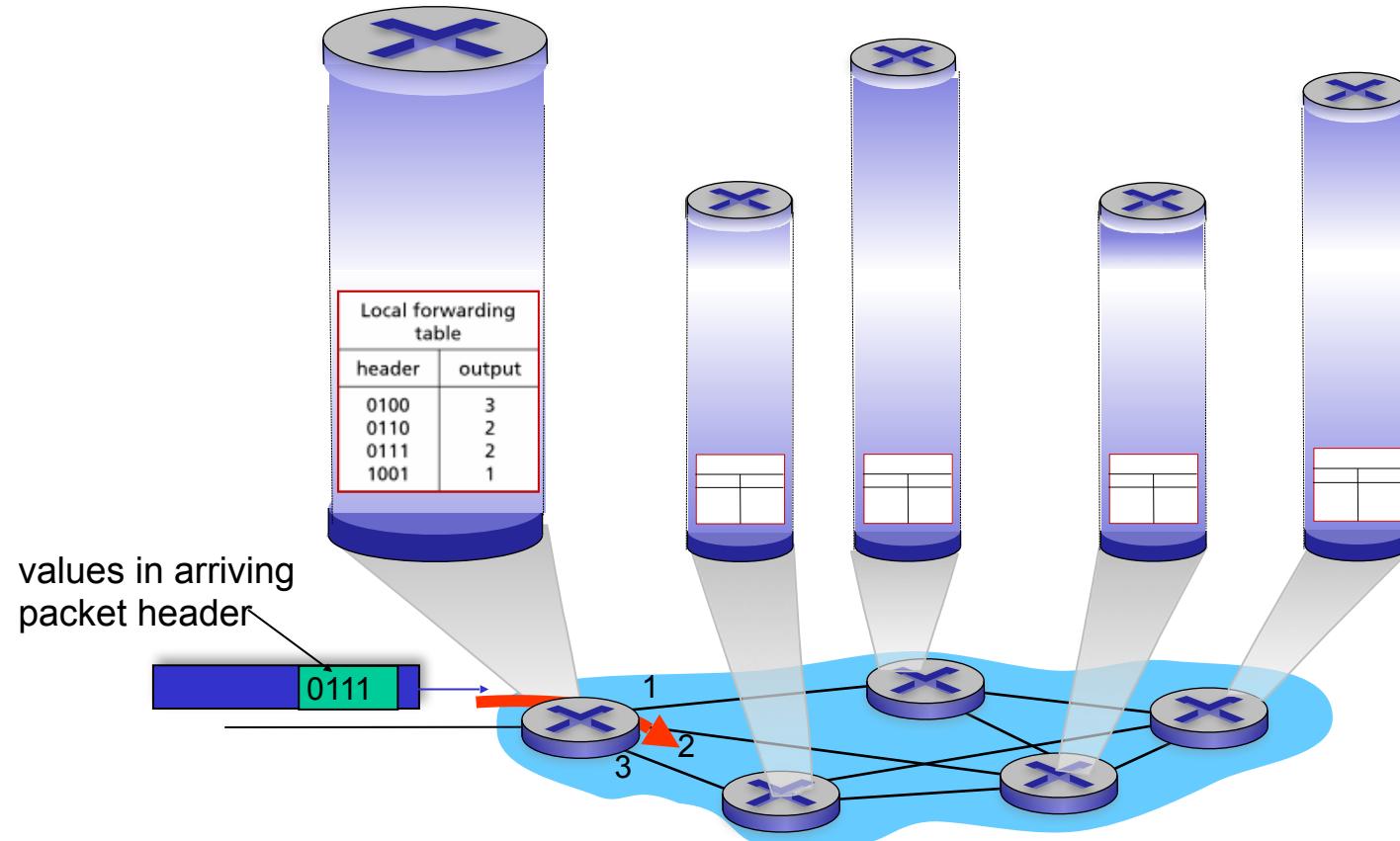
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



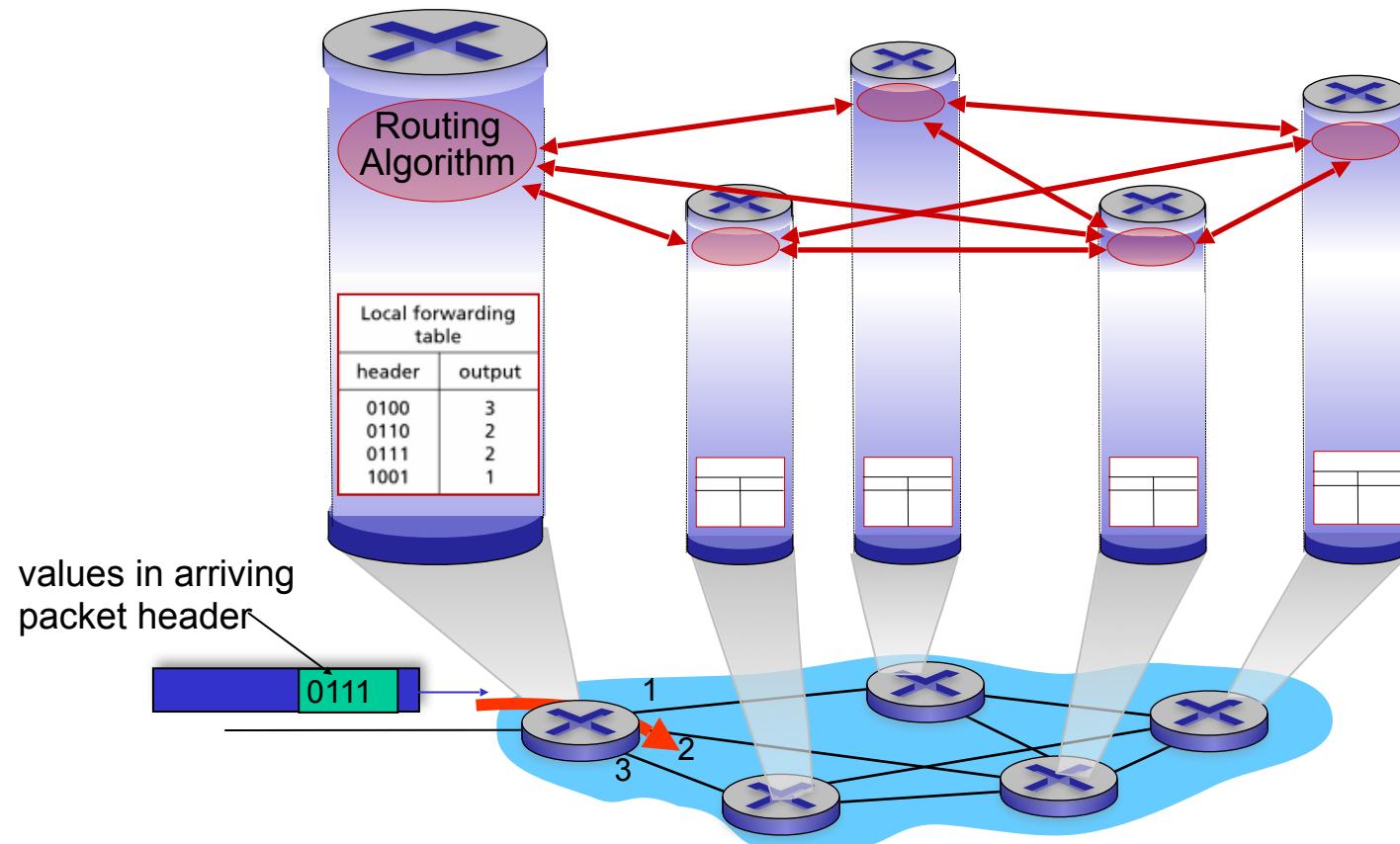
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



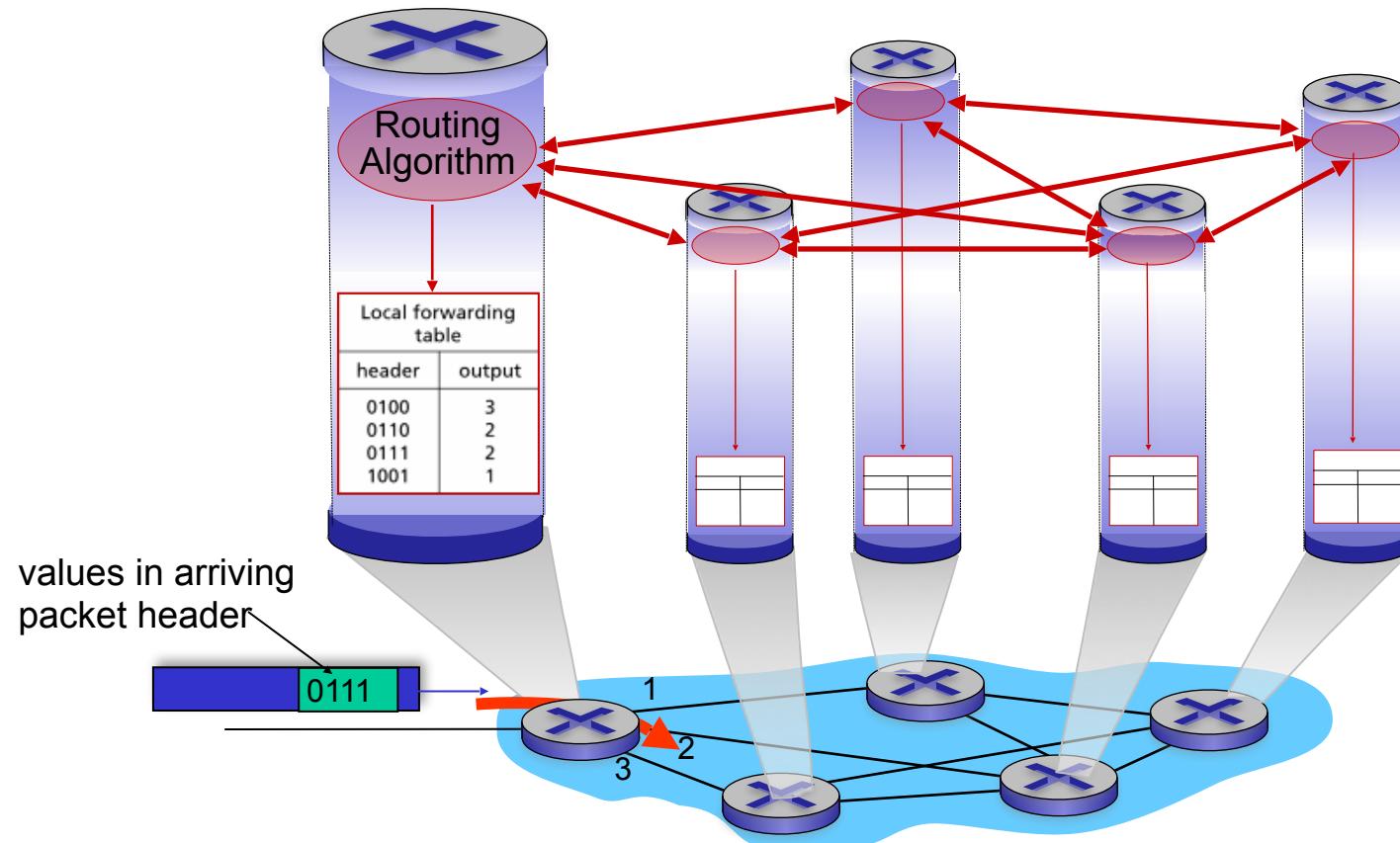
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



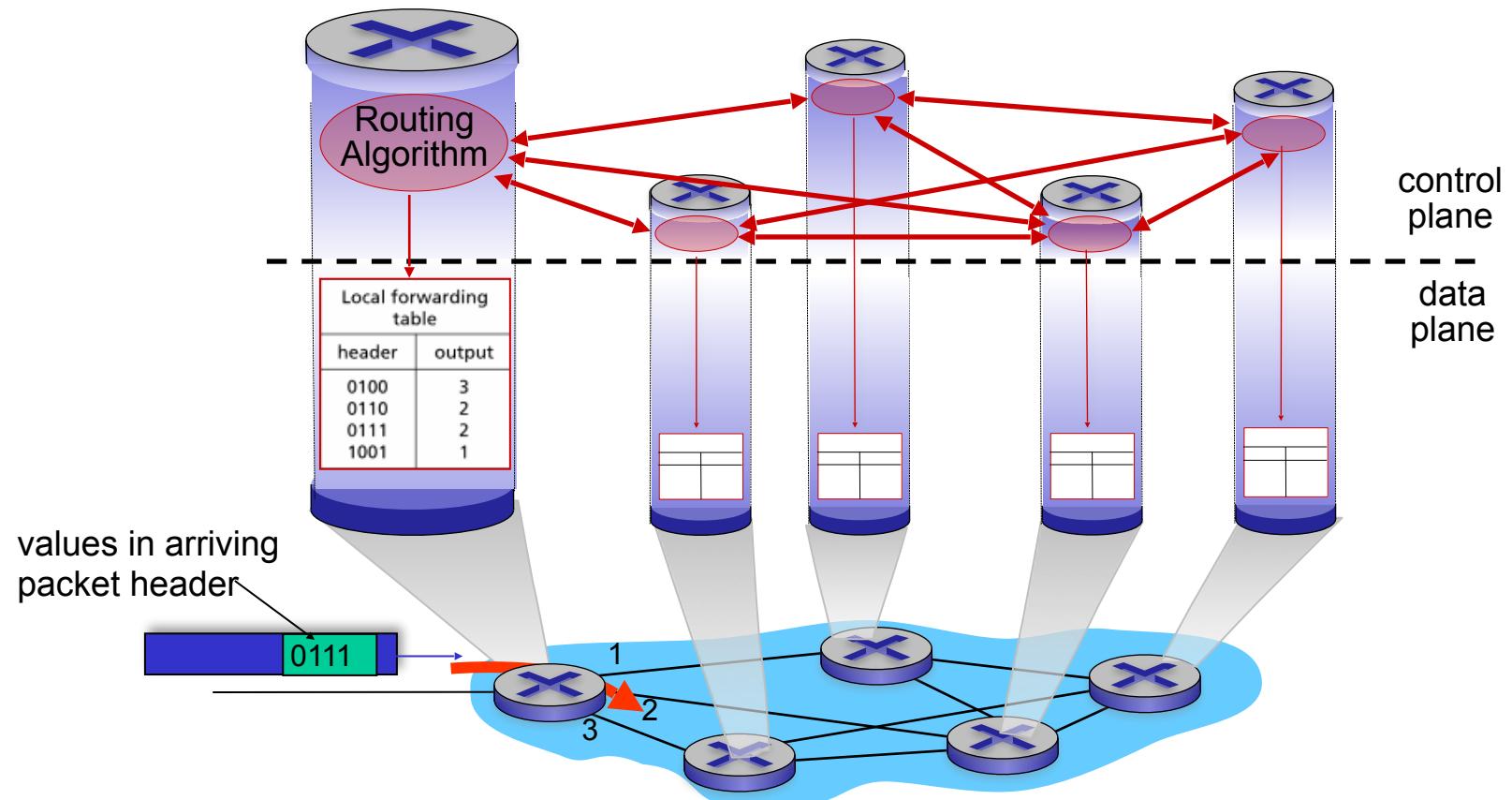
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



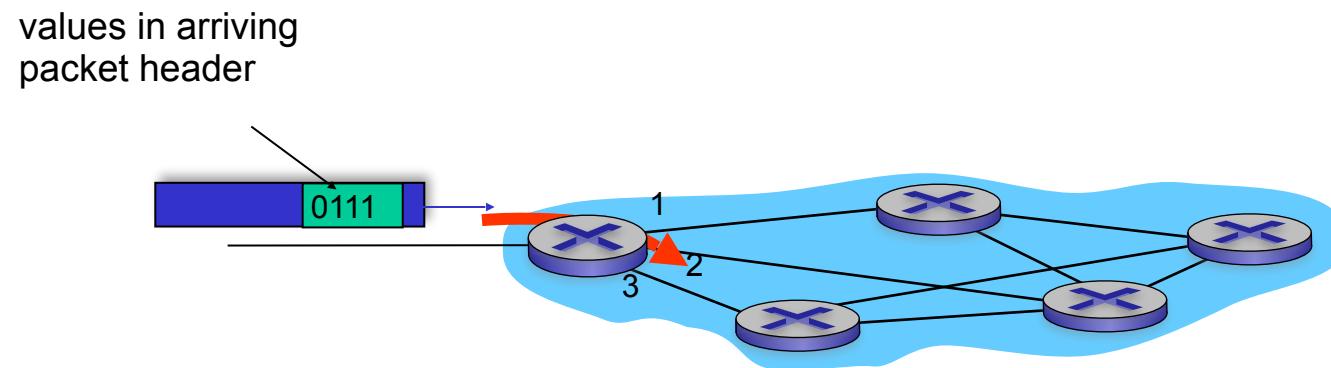
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane



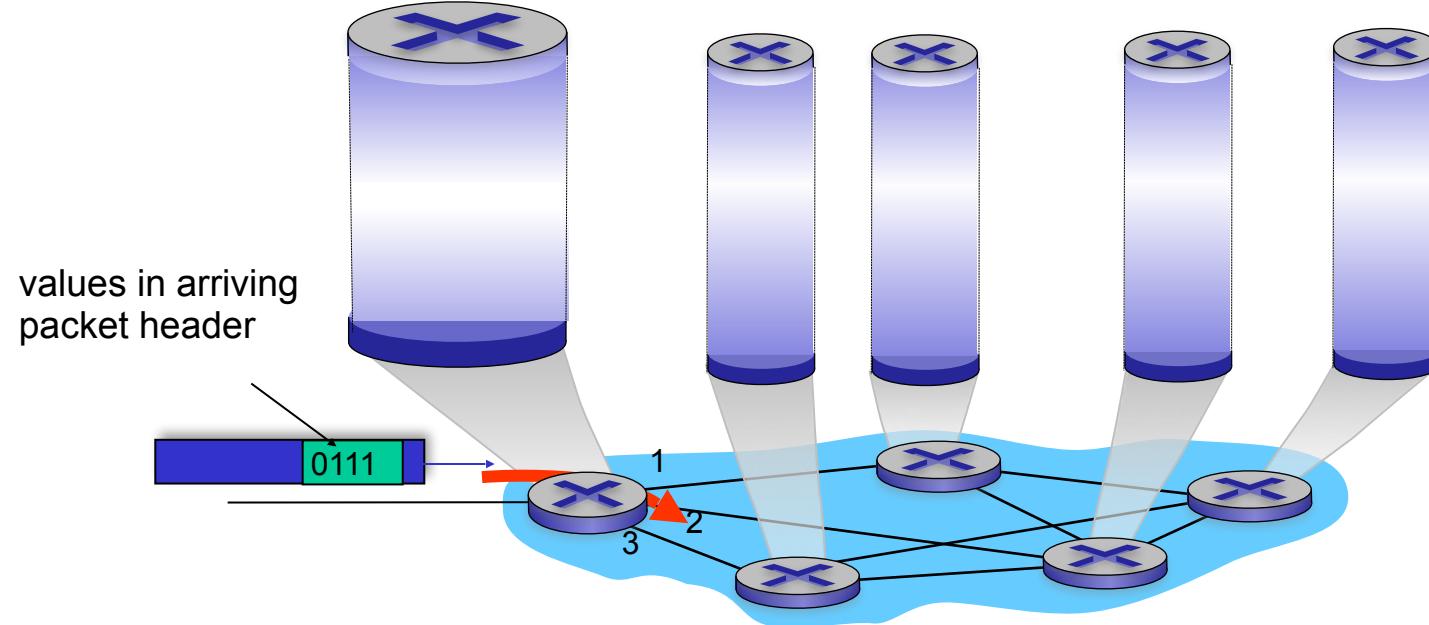
Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



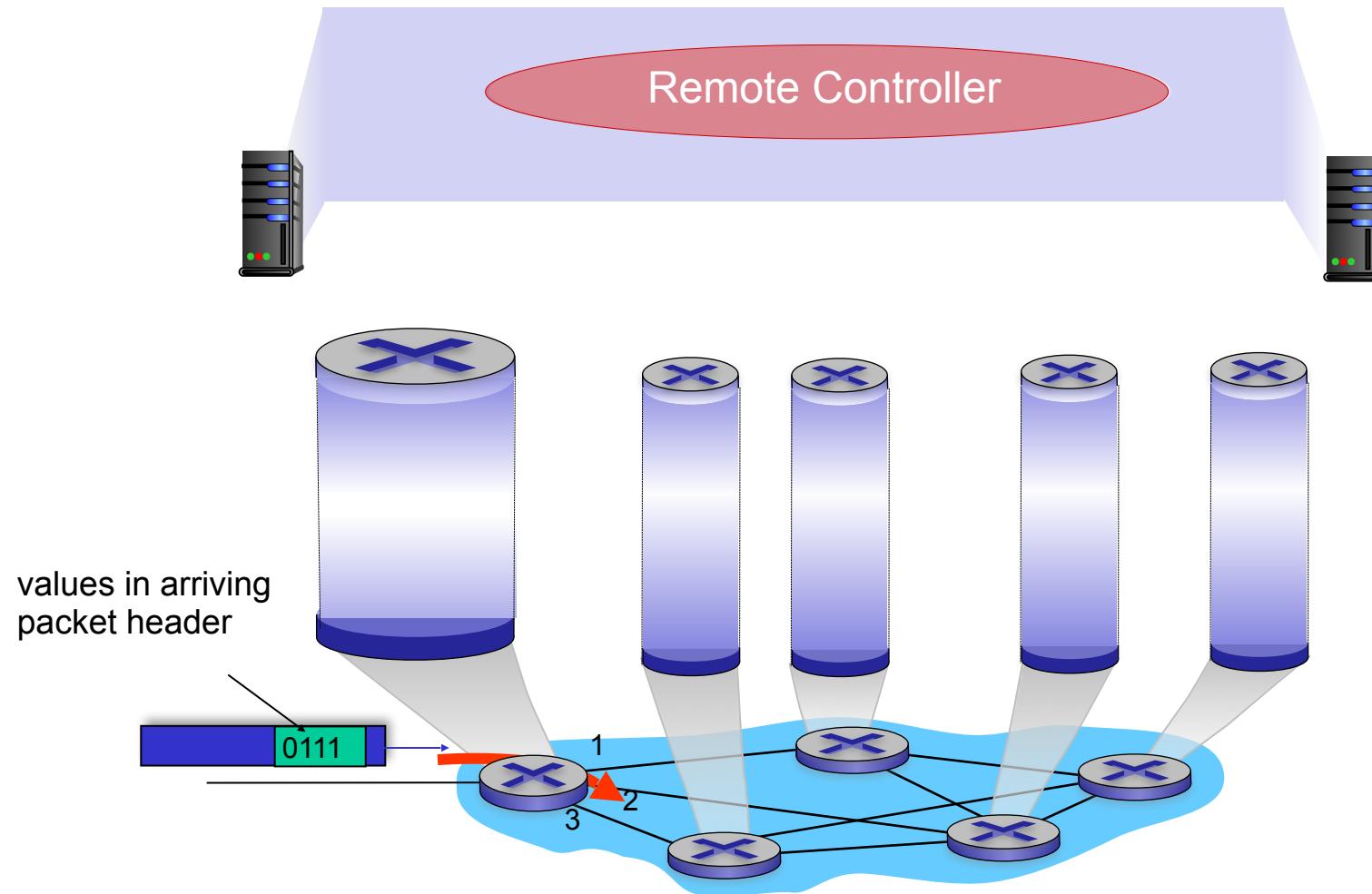
Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



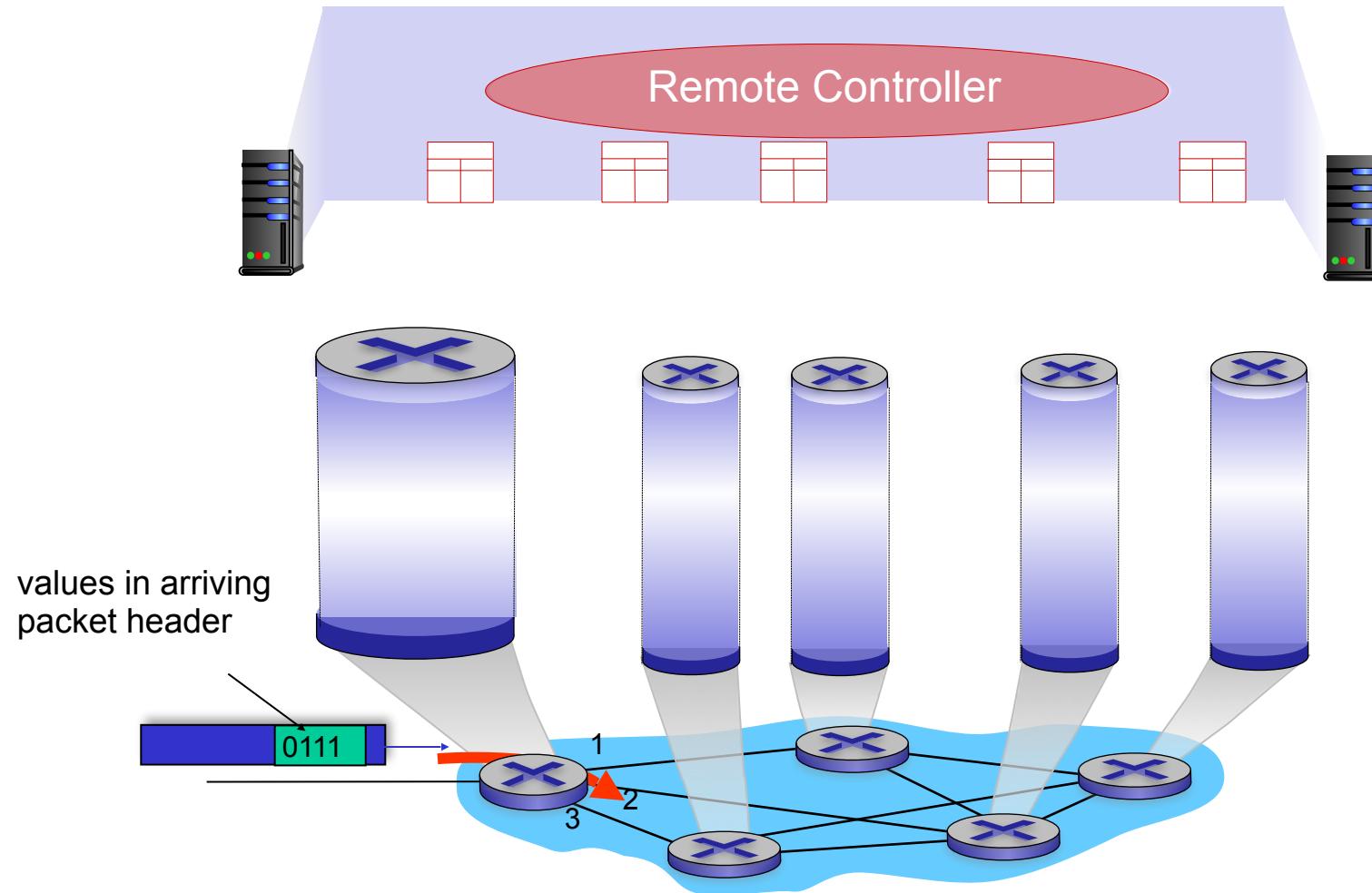
Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



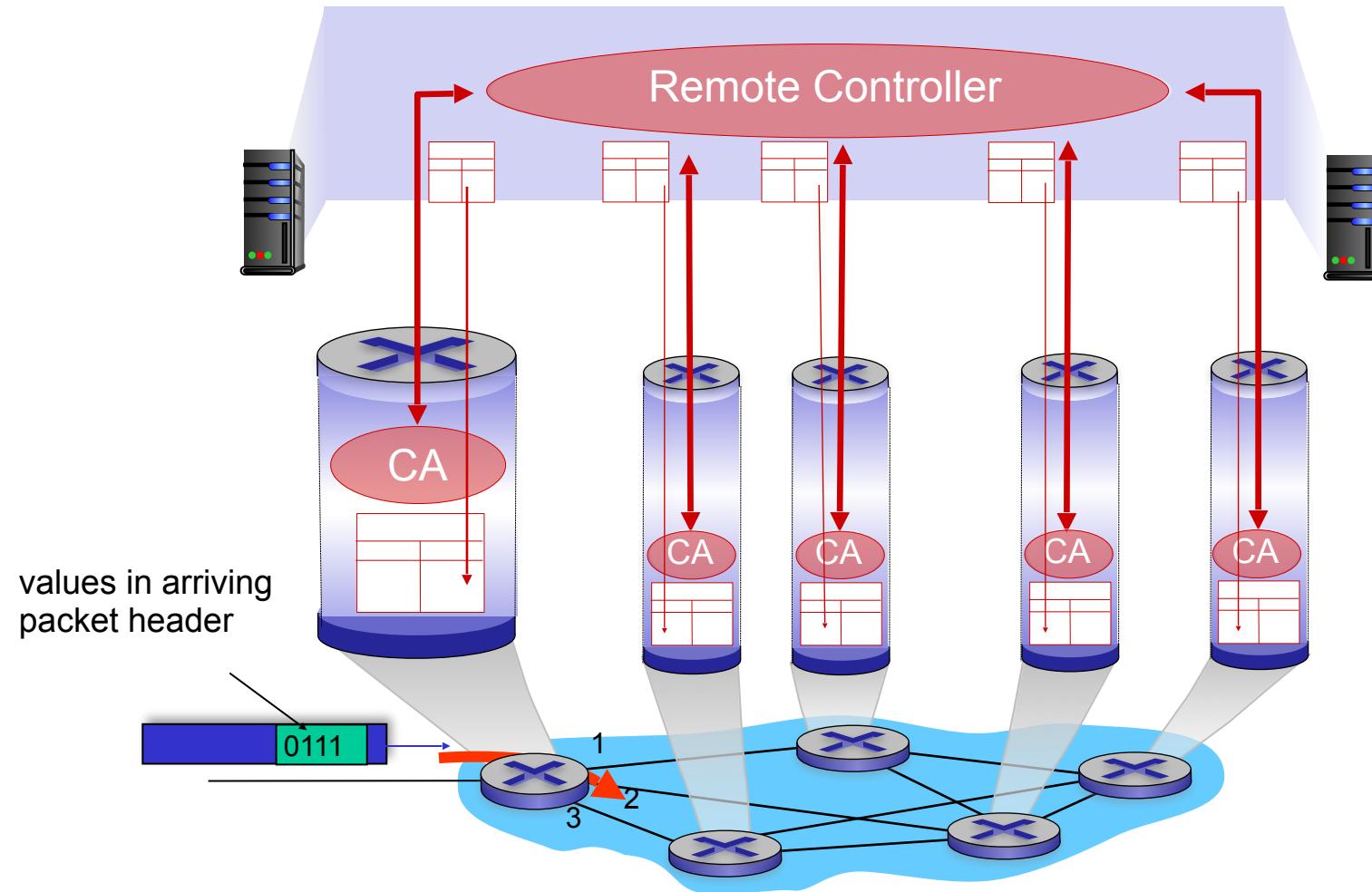
Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



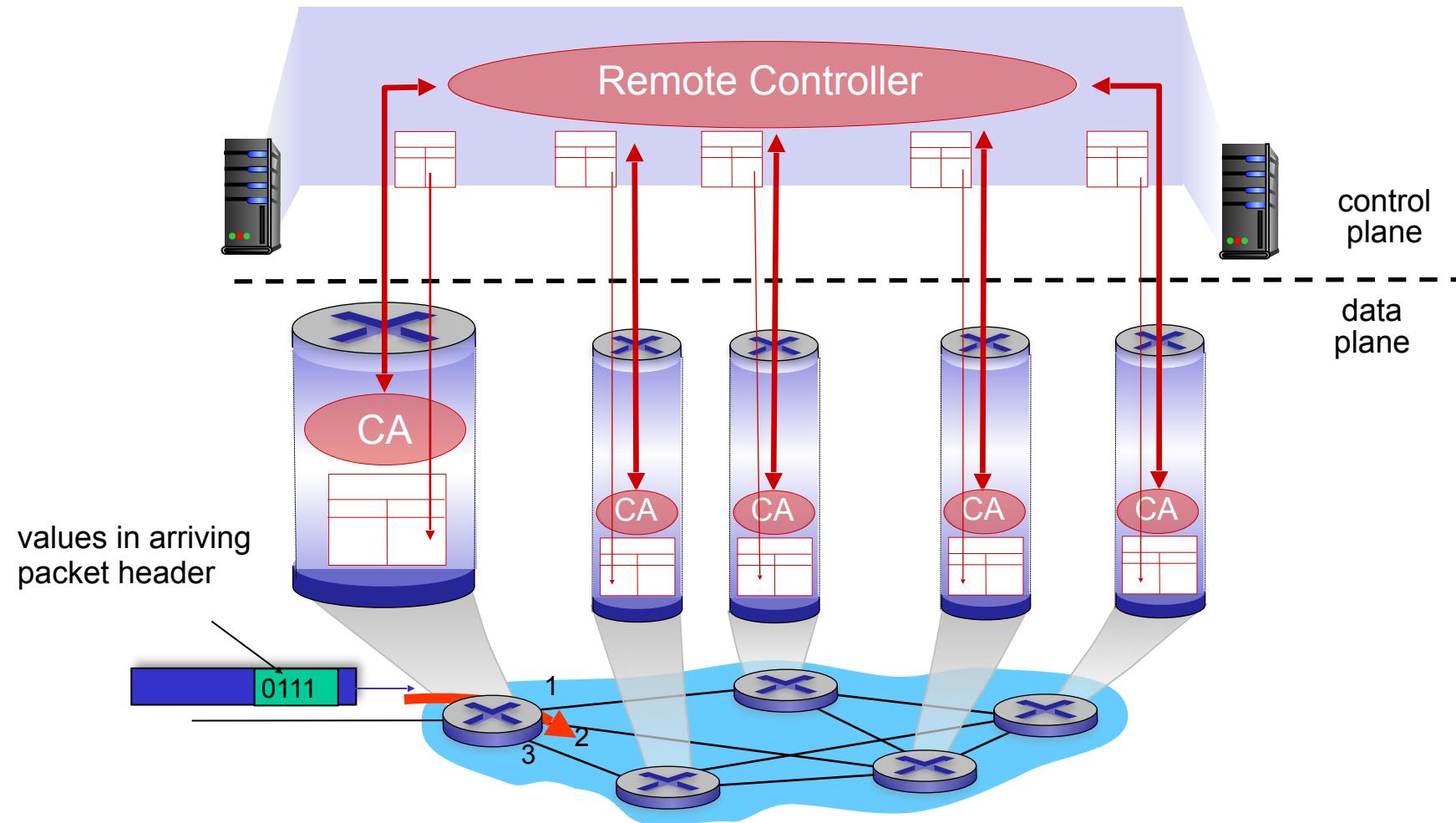
Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



Logically centralized control plane

Remote controller computes, installs forwarding tables in routers



Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for
individual datagrams:

Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for
individual datagrams:

example services for a *flow* of
datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for
individual datagrams:

- guaranteed delivery

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network service model

Q: What *service model* for “channel” transporting datagrams from sender to receiver?

example services for
individual datagrams:

- guaranteed delivery
- guaranteed delivery with less than 40 msec delay

example services for a *flow* of datagrams:

- in-order datagram delivery
- guaranteed minimum bandwidth to flow
- restrictions on changes in inter-packet spacing

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no

Internet “best effort” service model

No guarantees on:

- i. successful datagram delivery to destination
- ii. timing or order of delivery
- iii. bandwidth available to end-end flow

Network-layer service model

Network Architecture	Service Model	Quality of Service (QoS) Guarantees ?			
		Bandwidth	Loss	Order	Timing
Internet	best effort	none	no	no	no
ATM	Constant Bit Rate	Constant rate	yes	yes	yes
ATM	Available Bit Rate	Guaranteed min	no	yes	no
Internet	Intserv Guaranteed (RFC 1633)	yes	yes	yes	yes
Internet	Diffserv (RFC 2475)	possible	possibly	possibly	no

Reflections on best-effort service:

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

Reflections on best-effort service:

- simplicity of mechanism has allowed Internet to be widely deployed adopted
- sufficient provisioning of bandwidth allows performance of real-time applications (e.g., interactive voice, video) to be “good enough” for “most of the time”
- replicated, application-layer distributed services (datacenters, content distribution networks) connecting close to clients’ networks, allow services to be provided from multiple locations
- congestion control of “elastic” services helps

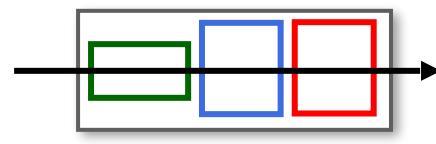
It's hard to argue with success of best-effort service model

Chapter 4: roadmap

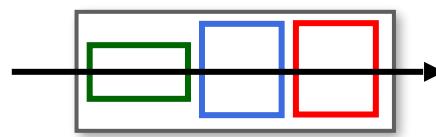
- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

Router architecture overview

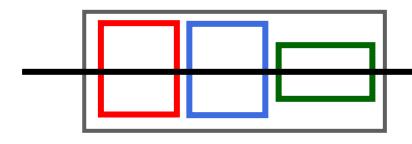
high-level view of generic router architecture:



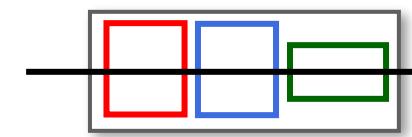
⋮



router input ports



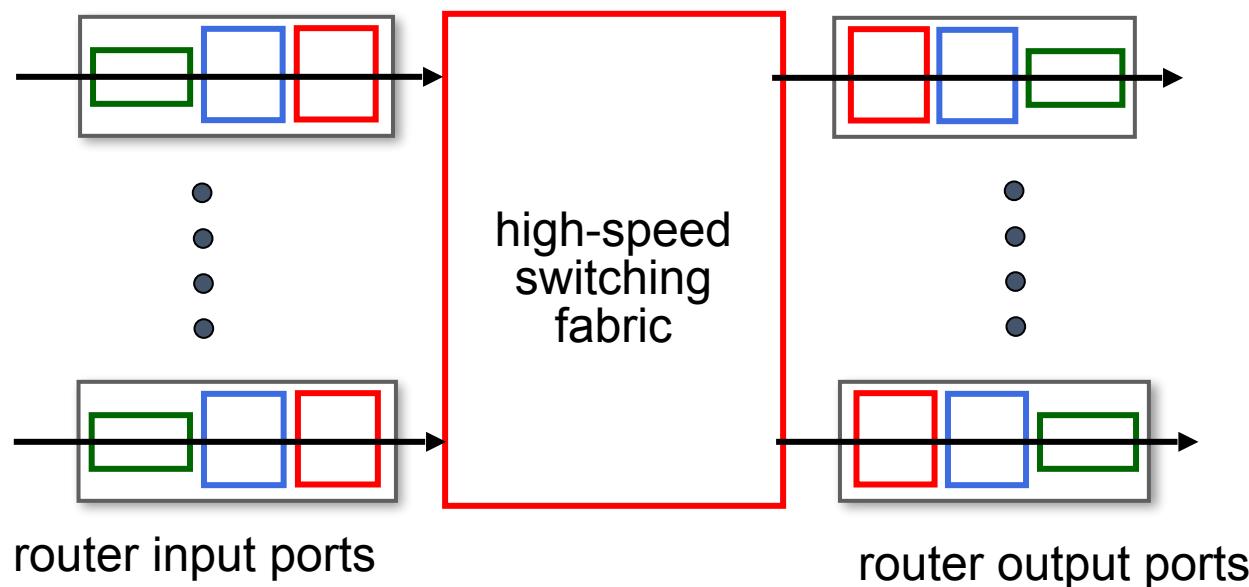
⋮



router output ports

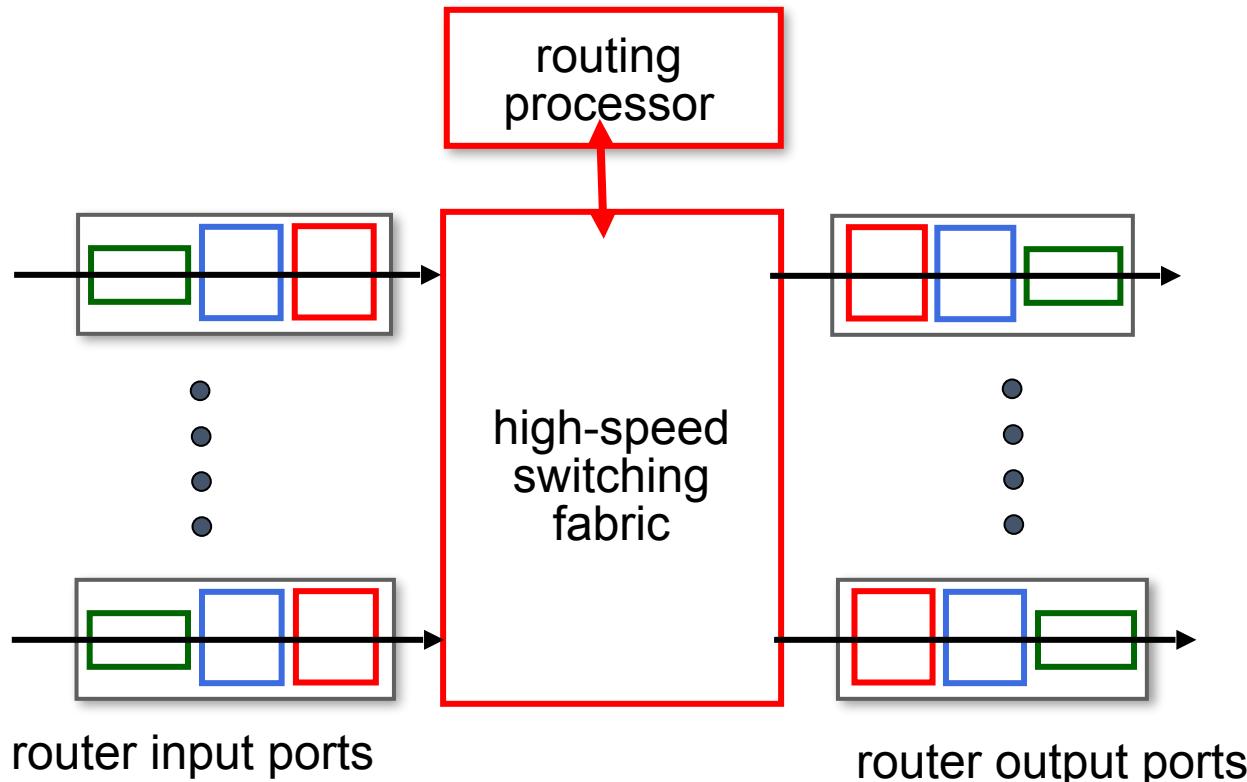
Router architecture overview

high-level view of generic router architecture:



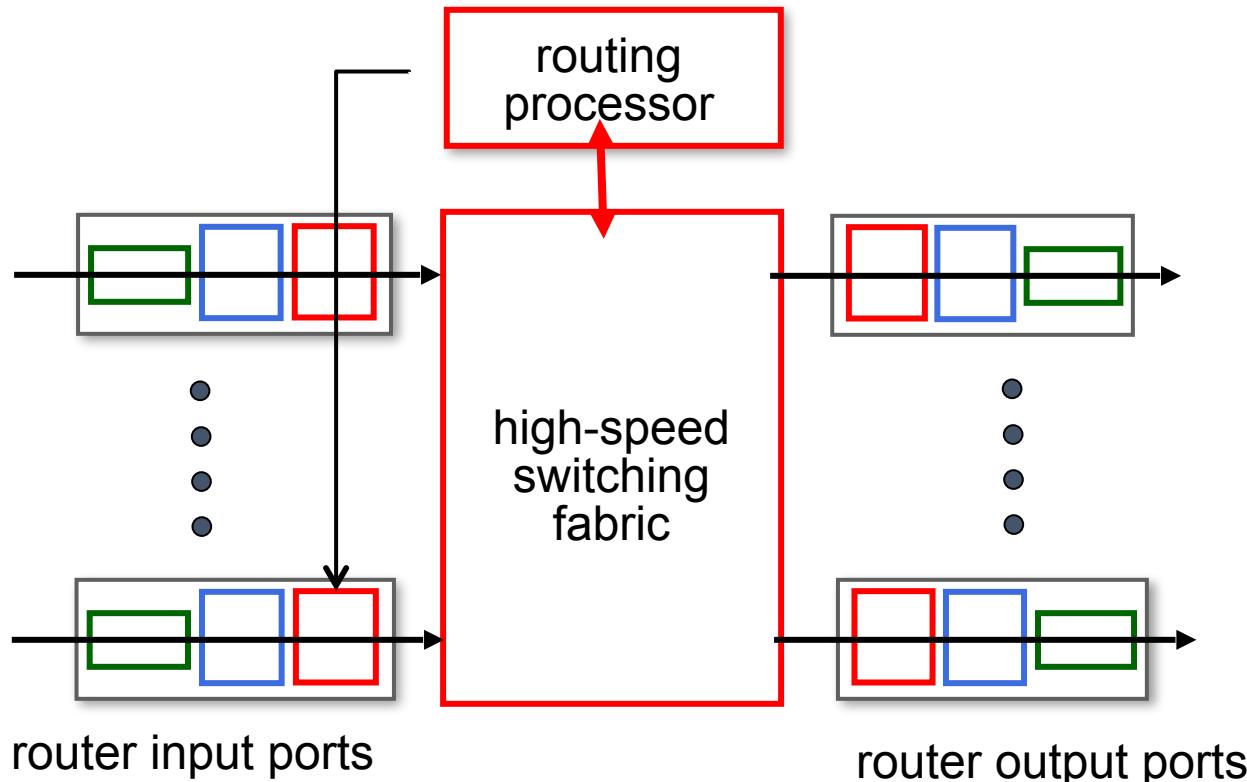
Router architecture overview

high-level view of generic router architecture:



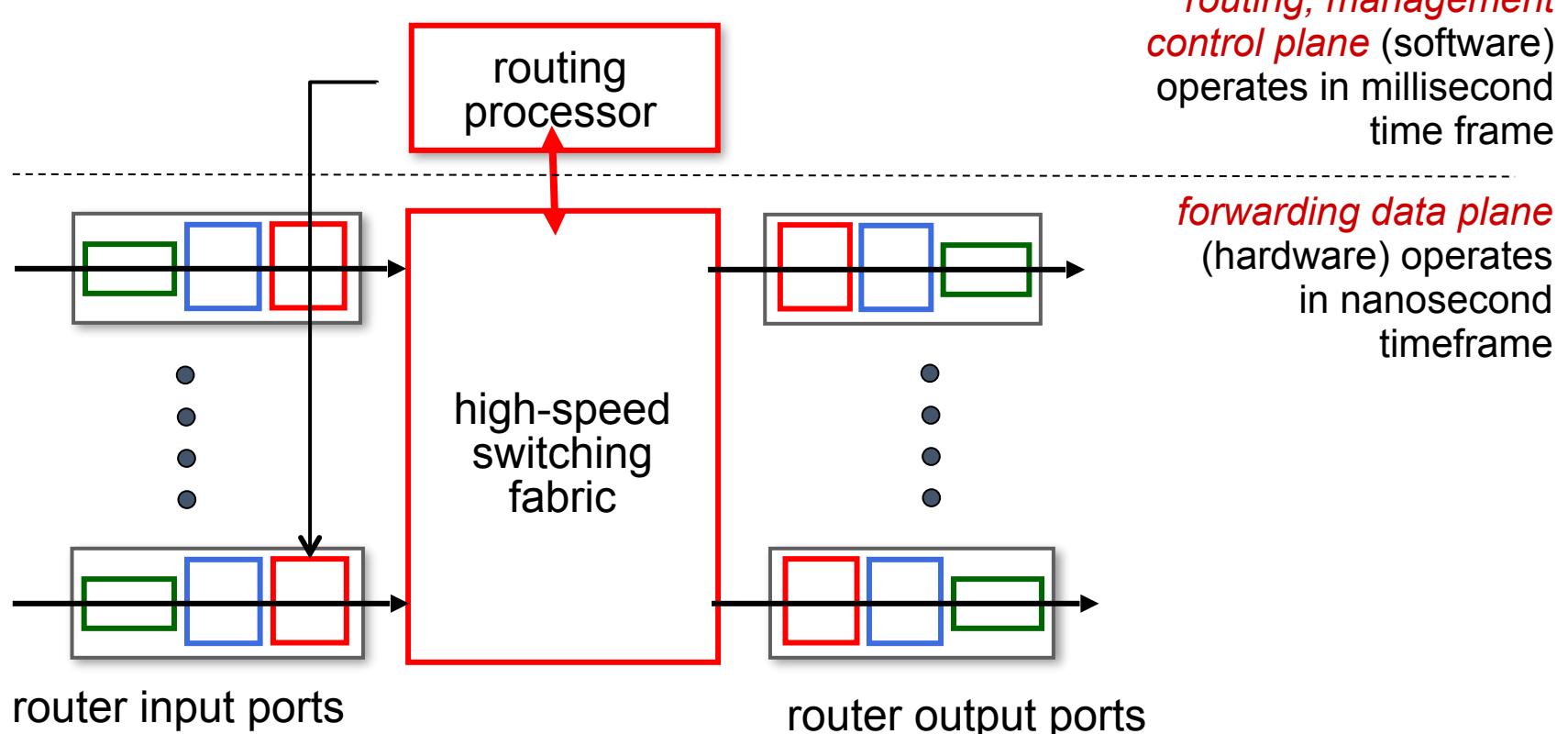
Router architecture overview

high-level view of generic router architecture:



Router architecture overview

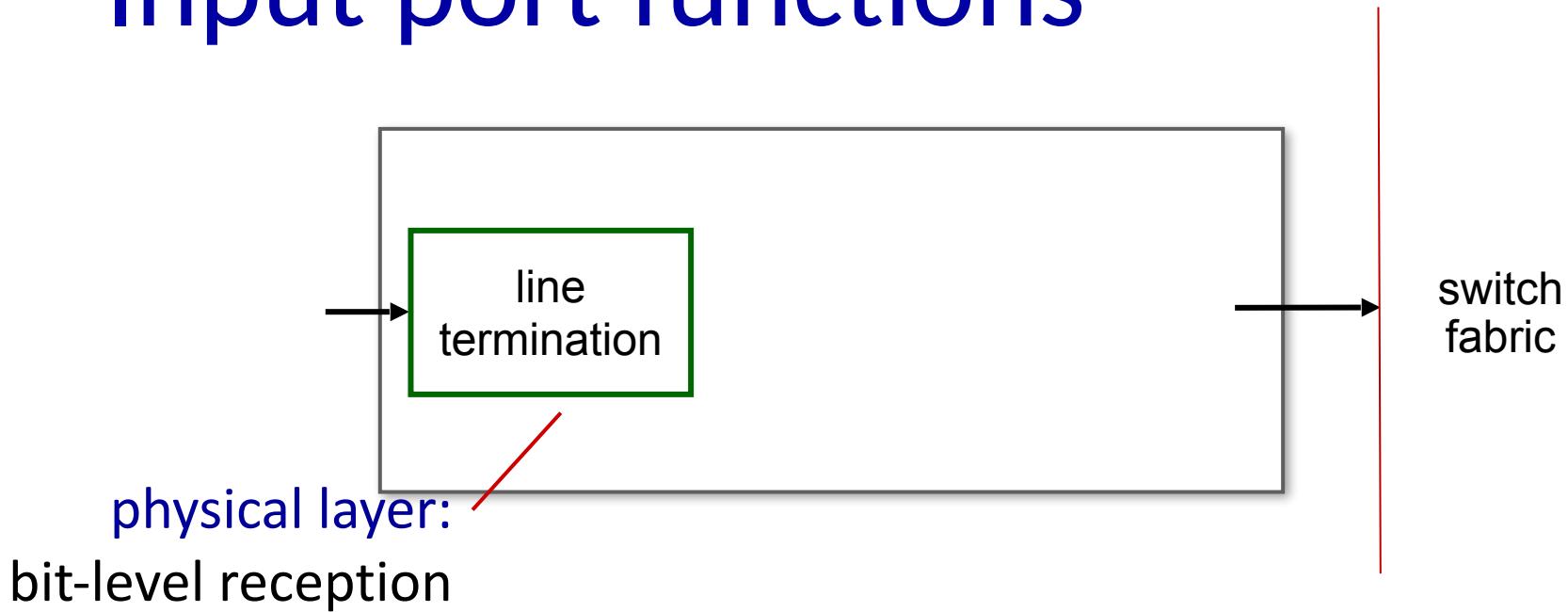
high-level view of generic router architecture:



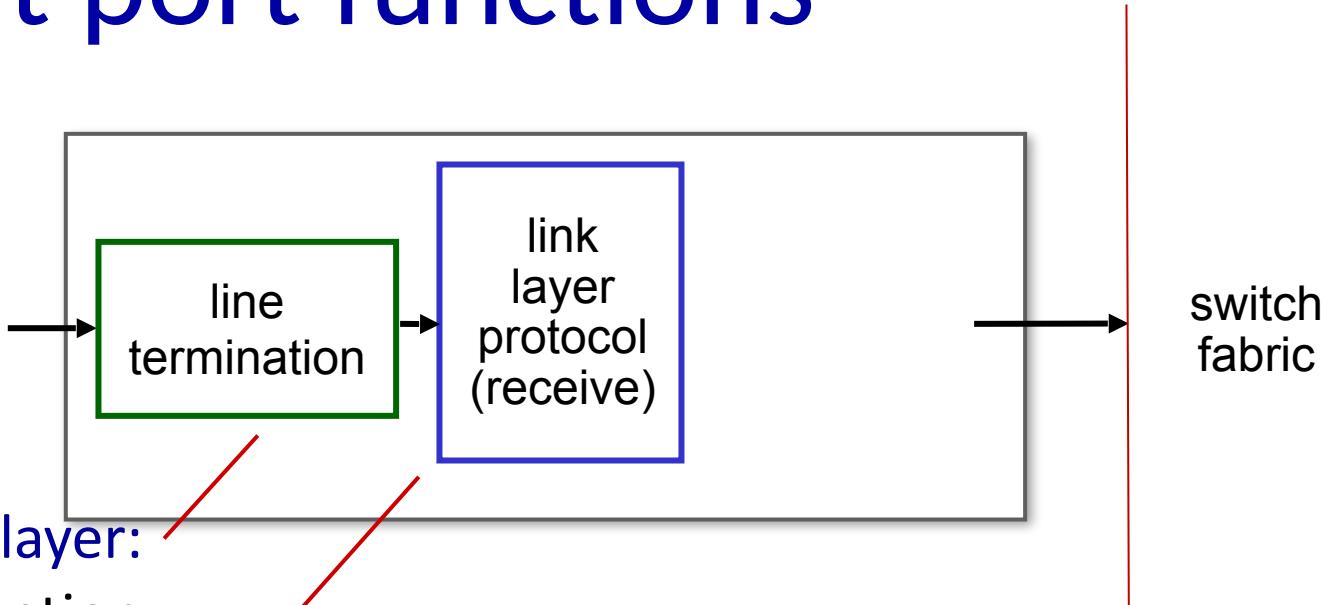
Input port functions



Input port functions



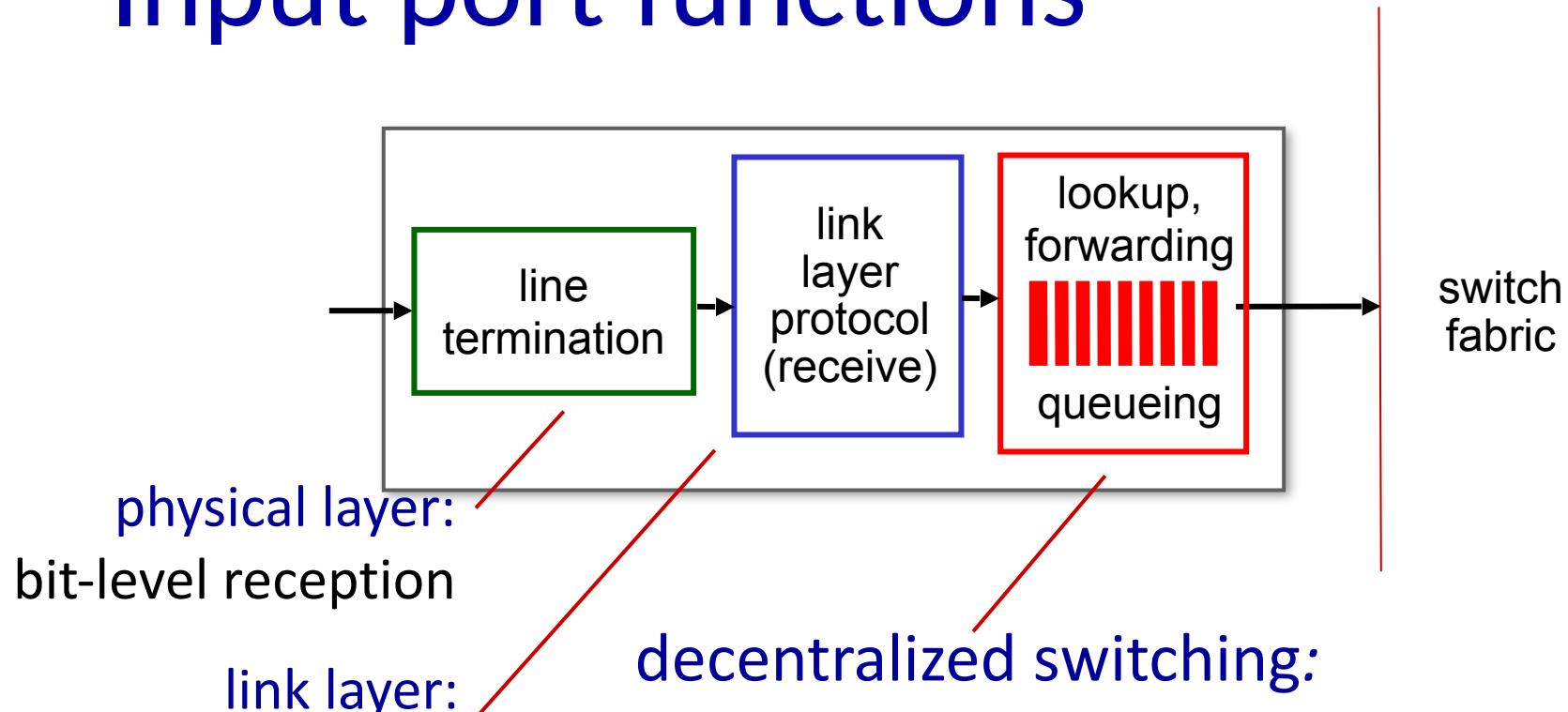
Input port functions



physical layer:
bit-level reception

link layer:
e.g., Ethernet
(chapter 6)

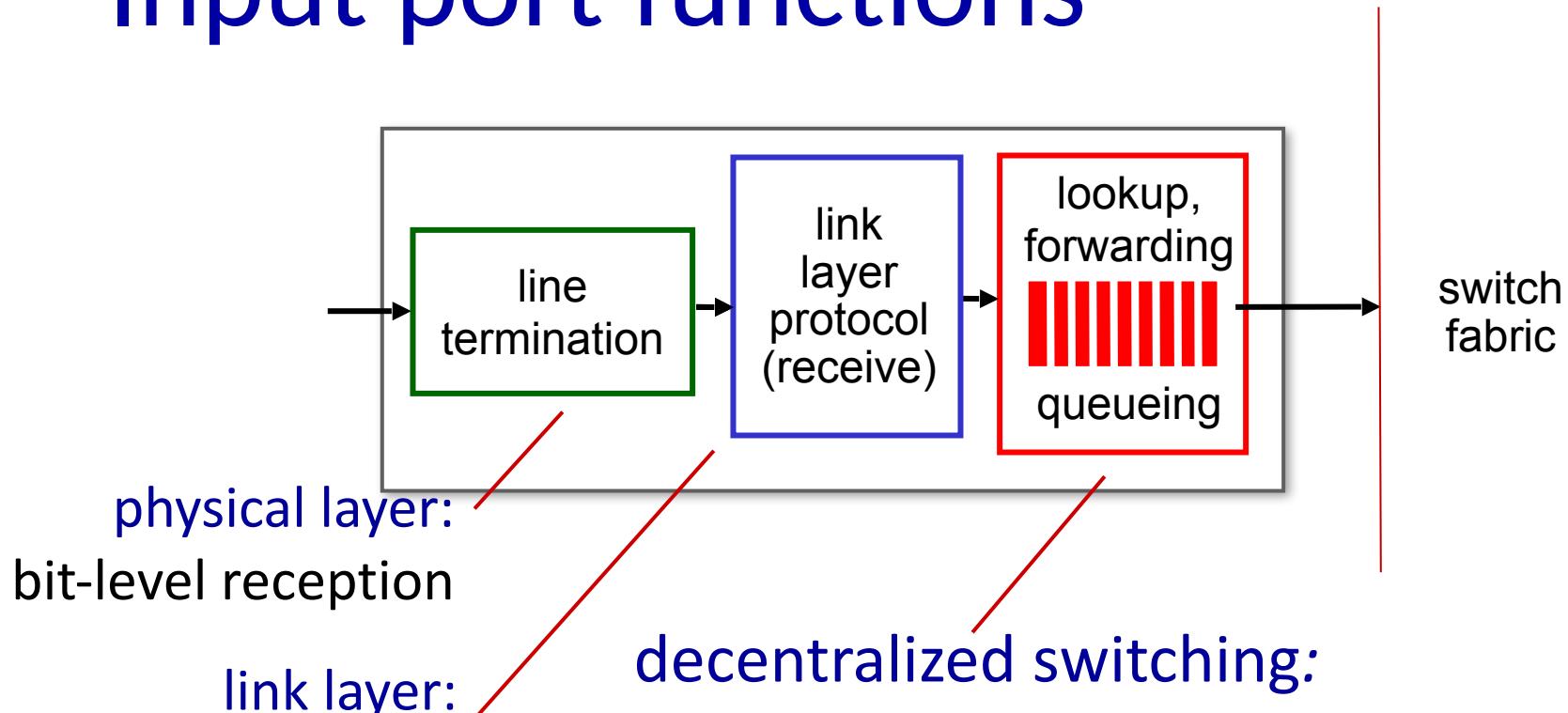
Input port functions



decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory (*"match plus action"*)
- goal: complete input port processing at 'line speed'
- **input port queuing:** if datagrams arrive faster than forwarding rate into switch fabric

Input port functions



decentralized switching:

- using header field values, lookup output port using forwarding table in input port memory (*"match plus action"*)
- **destination-based forwarding**: forward based only on destination IP address (traditional)
- **generalized forwarding**: forward based on any set of header field values

Destination-based forwarding

<i>forwarding table</i>	
Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Q: but what happens if ranges don't divide up so nicely?

Destination-based forwarding

forwarding table

Destination Address Range	Link Interface
11001000 00010111 00010000 00000000 through 11001000 00010111 00010111 11111111	0
11001000 00010111 00010000 00000100 through 11001000 00010111 00010000 00000111	3
11001000 00010111 00011000 00000000 through 11001000 00010111 00011000 11111111	1
11001000 00010111 00011001 00000000 through 11001000 00010111 00011111 11111111	2
otherwise	3

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011*** *****	2
otherwise	3

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011_*_* *****	2
otherwise	3

examples:

- | | |
|-------------------------------------|------------------|
| 11001000 00010111 00010110 10100001 | which interface? |
| 11001000 00010111 00011000 10101010 | which interface? |

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011_*_* *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range			Link interface	
11001000	00010111	00010*****	0	
11001000	00010111	00011000*****	1	
11001000	00010111	00011*****	2	
otherwise			3	

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011_*_* *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	*****	0
11001000	00010111	00011000	*****	1
11001000	00010111	00011***	*****	2
otherwise				3

match!

examples:

11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range	Link interface
11001000 00010111 00010*** *****	0
11001000 00010111 00011000 *****	1
11001000 00010111 00011_*_* *****	2
otherwise	3

examples:

11001000 00010111 00010110 10100001 which interface?

11001000 00010111 00011000 10101010 which interface?

Longest prefix matching

longest prefix match

when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.

Destination Address Range				Link interface
11001000	00010111	00010***	*****	0
11001000	00010111	00011000	*****	1
11001000	00010111	00011***	*****	2
otherwise				3

match!

examples:

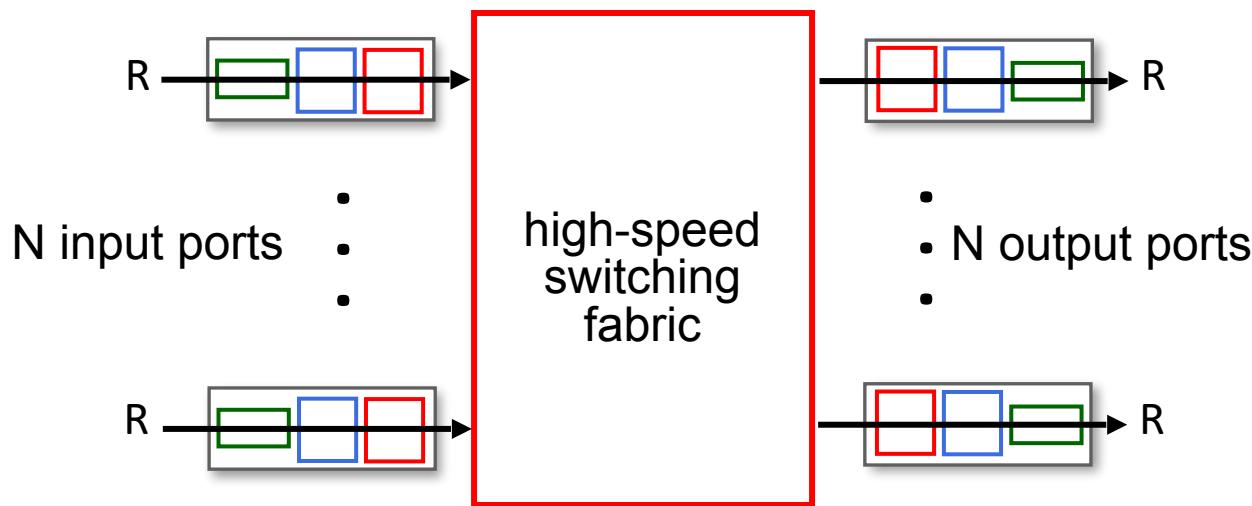
11001000	00010111	00010110	10100001	which interface?
11001000	00010111	00011000	10101010	which interface?

Longest prefix matching

- we'll see *why* longest prefix matching is used shortly, when we study addressing
- longest prefix matching: often performed using ternary content addressable memories (TCAMs)
 - *content addressable*: present address to TCAM: retrieve address in one clock cycle, regardless of table size
 - Cisco Catalyst: ~1M routing table entries in TCAM

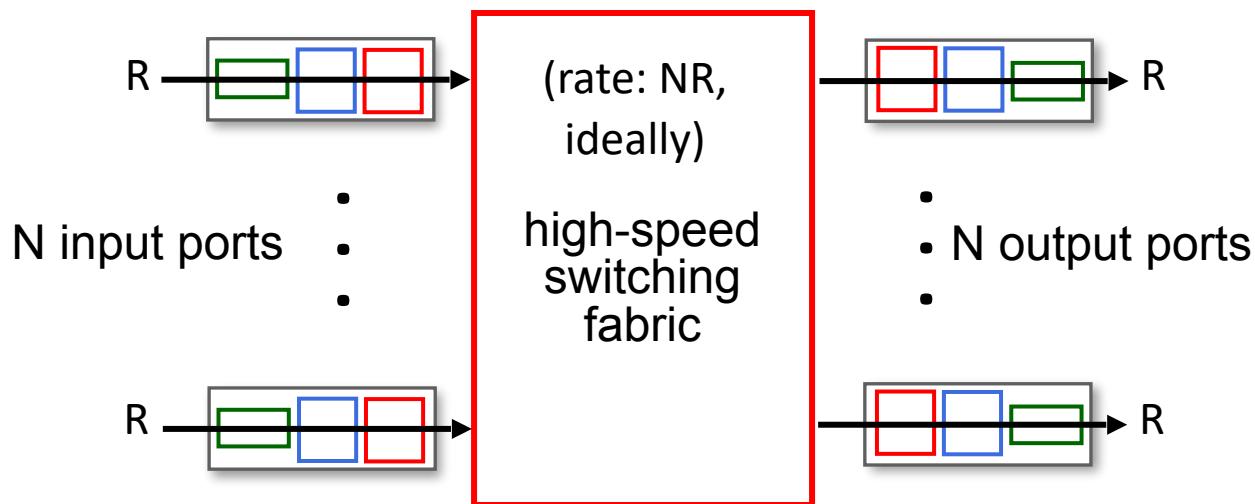
Switching fabrics

- transfer packet from input link to appropriate output link



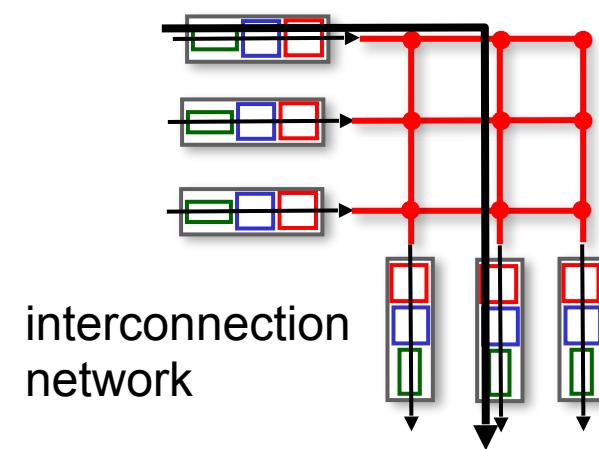
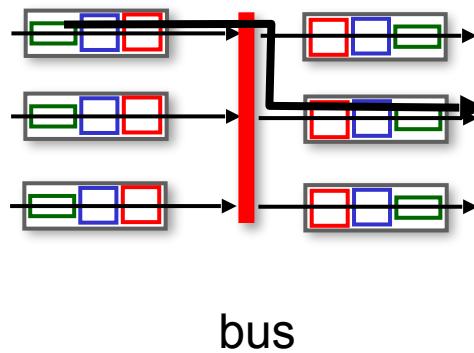
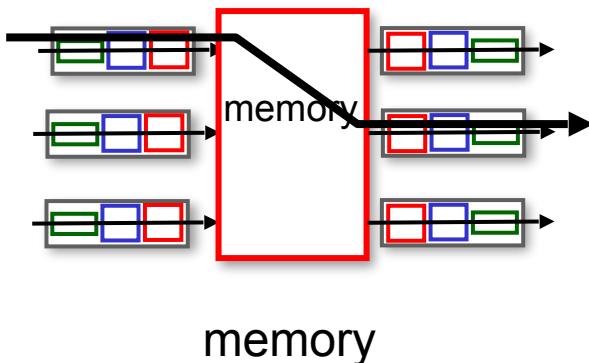
Switching fabrics

- transfer packet from input link to appropriate output link
- **switching rate:** rate at which packets can be transferred from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable



Switching fabrics

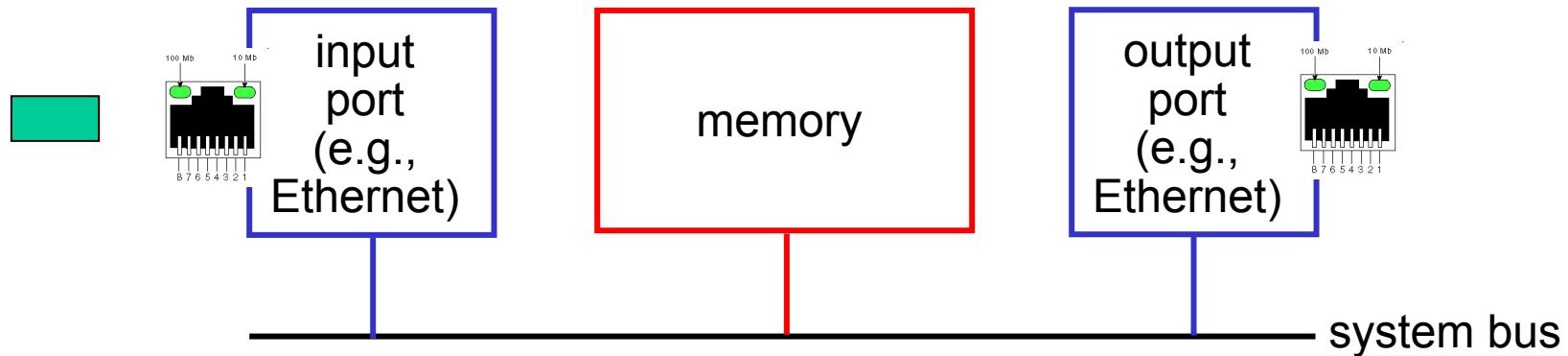
- transfer packet from input link to appropriate output link
- **switching rate:** rate at which packets can be transferred from inputs to outputs
 - often measured as multiple of input/output line rate
 - N inputs: switching rate N times line rate desirable
- three major types of switching fabrics:



Switching via memory

first generation routers:

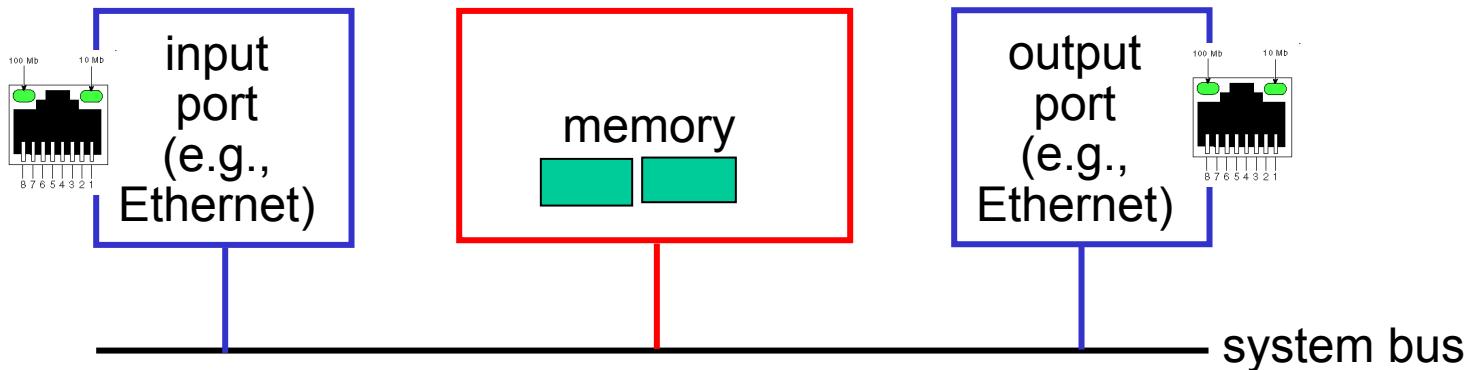
- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via memory

first generation routers:

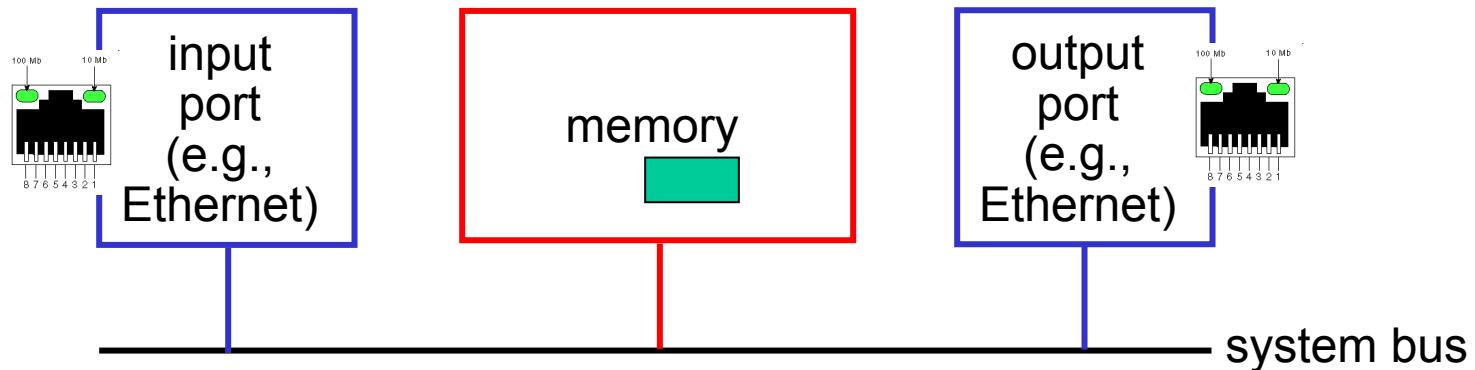
- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



Switching via memory

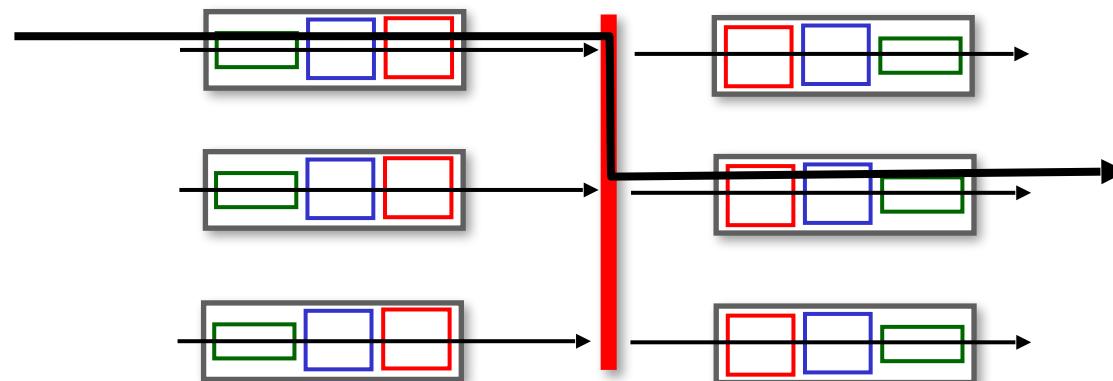
first generation routers:

- traditional computers with switching under direct control of CPU
- packet copied to system's memory
- speed limited by memory bandwidth (2 bus crossings per datagram)



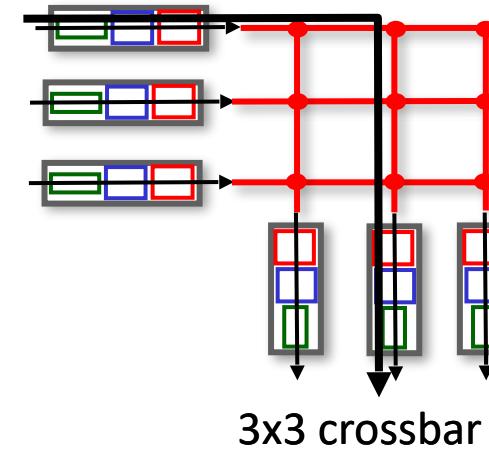
Switching via a bus

- datagram from input port memory to output port memory via a shared bus
- *bus contention*: switching speed limited by bus bandwidth
- 32 Gbps bus, Cisco 5600: sufficient speed for access routers



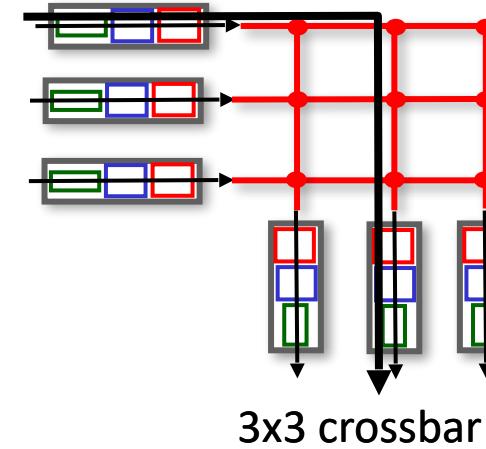
Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor

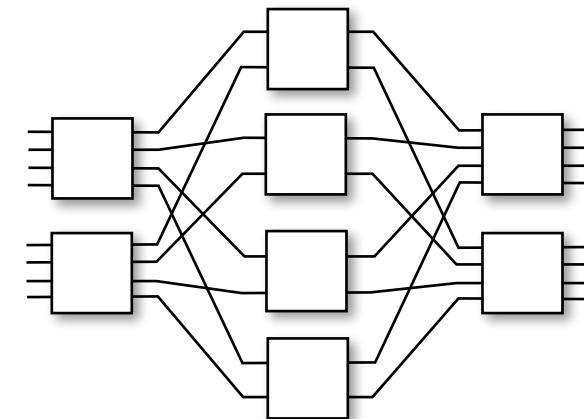


Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- **multistage switch:** nxn switch from multiple stages of smaller switches



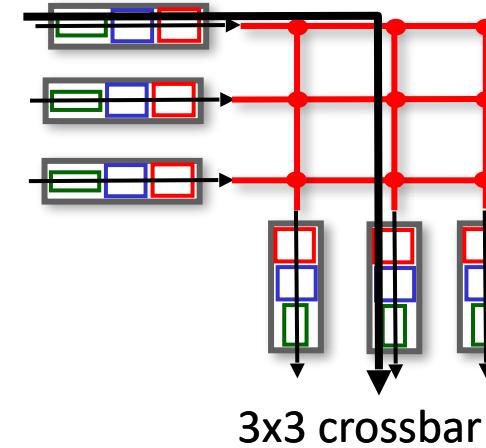
3x3 crossbar



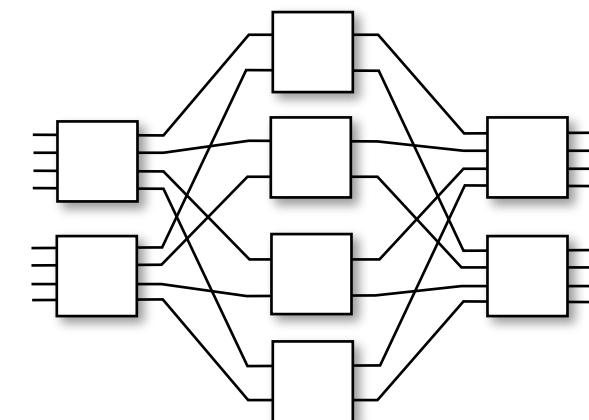
8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- **multistage switch:** nxn switch from multiple stages of smaller switches
- **exploiting parallelism:**
 - fragment datagram into fixed length cells on entry
 - switch cells through the fabric, reassemble datagram at exit



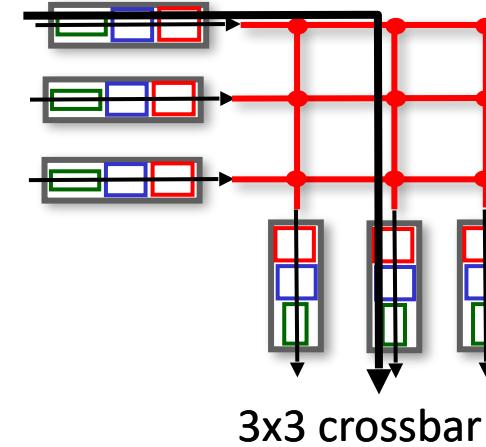
3x3 crossbar



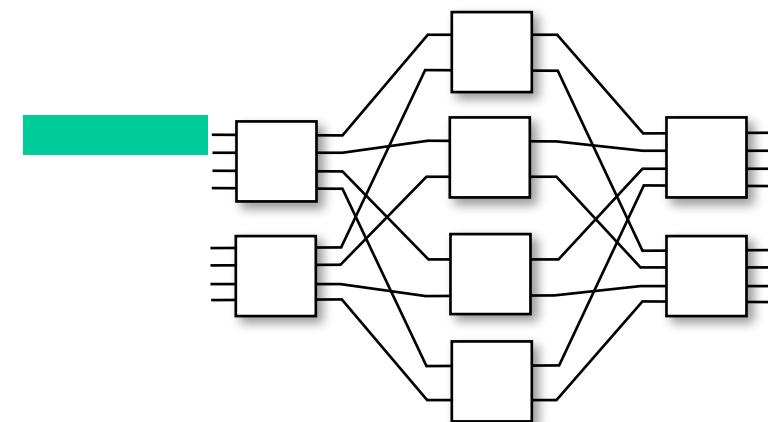
8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- **multistage switch:** nxn switch from multiple stages of smaller switches
- **exploiting parallelism:**
 - fragment datagram into fixed length cells on entry
 - switch cells through the fabric, reassemble datagram at exit



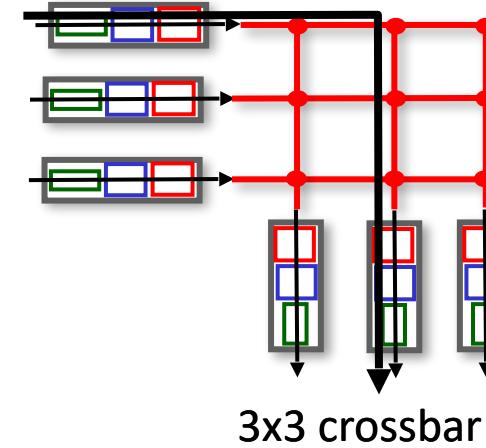
3x3 crossbar



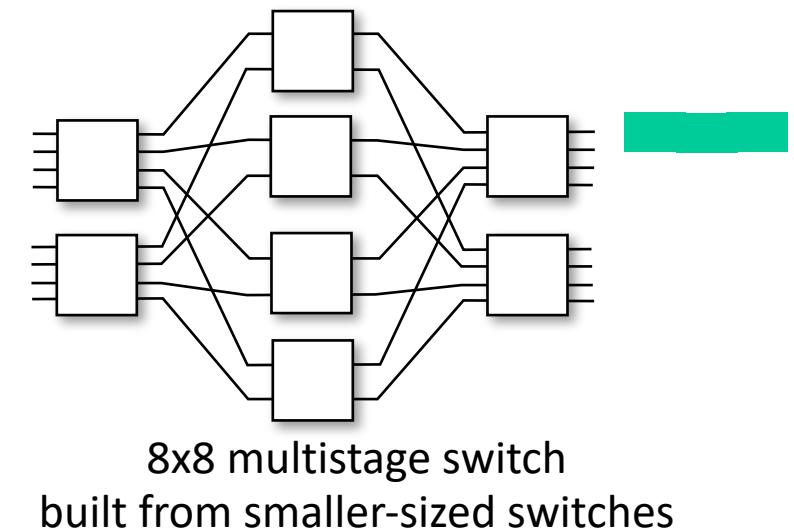
8x8 multistage switch
built from smaller-sized switches

Switching via interconnection network

- Crossbar, Clos networks, other interconnection nets initially developed to connect processors in multiprocessor
- **multistage switch:** nxn switch from multiple stages of smaller switches
- **exploiting parallelism:**
 - fragment datagram into fixed length cells on entry
 - switch cells through the fabric, reassemble datagram at exit



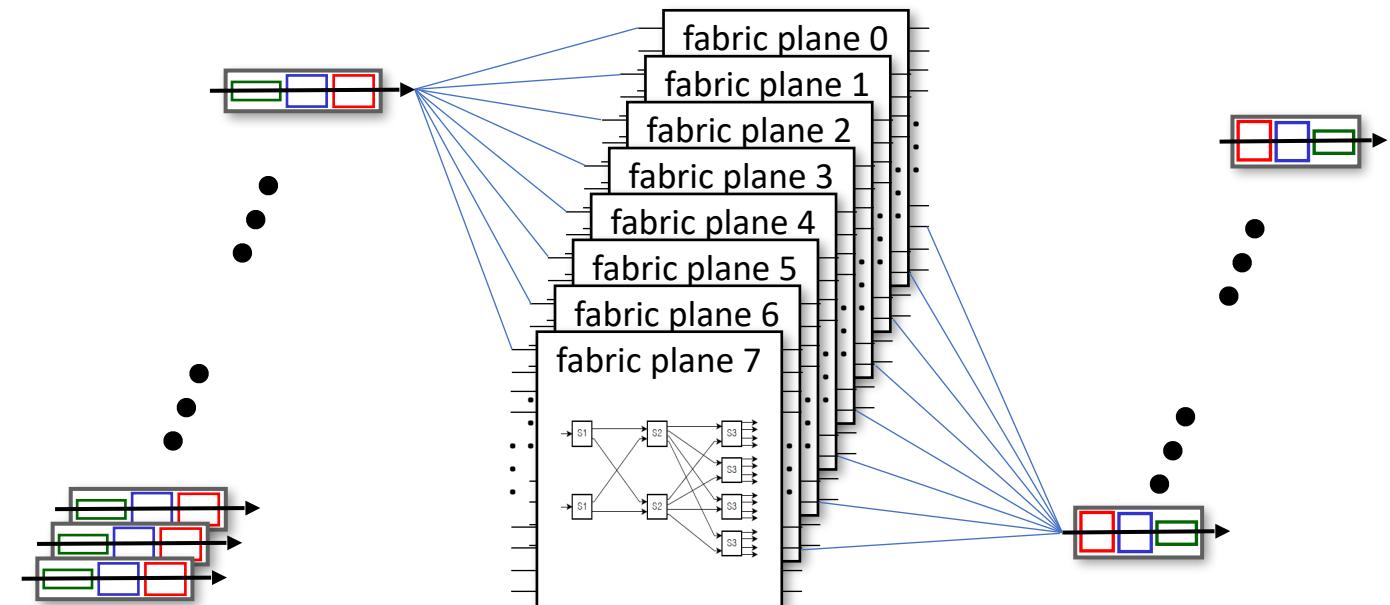
3x3 crossbar



8x8 multistage switch
built from smaller-sized switches

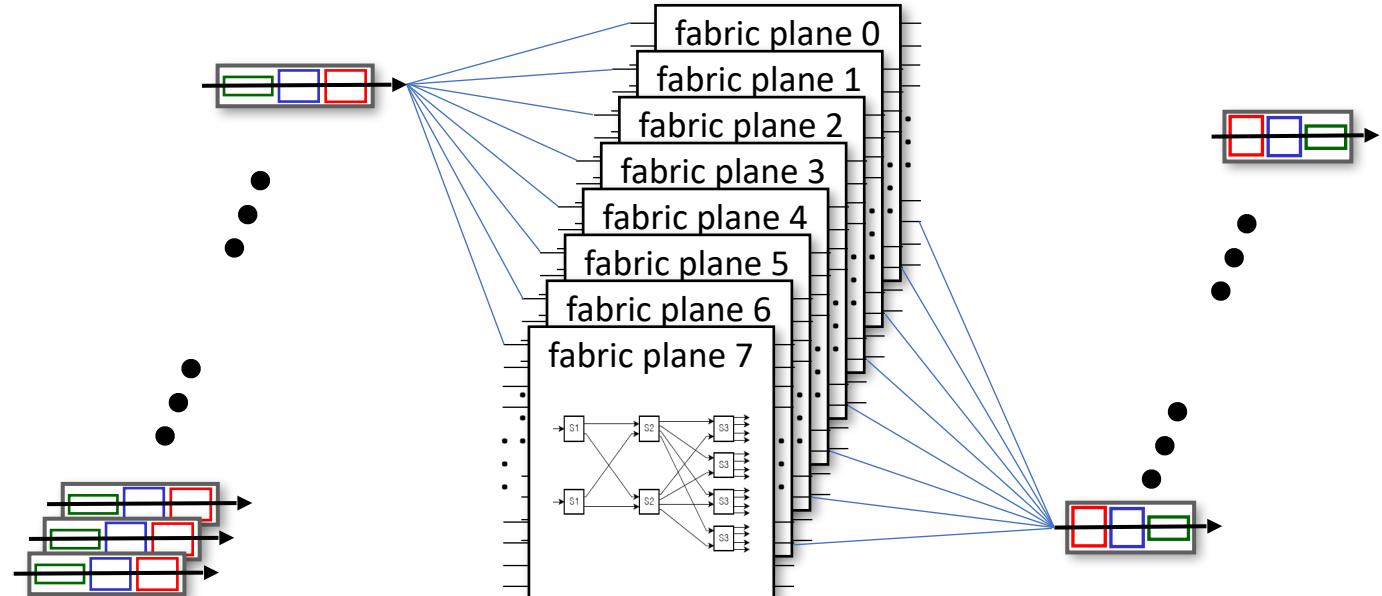
Switching via interconnection network

- scaling, using multiple switching “planes” in parallel:
 - speedup, scaleup via parallelism



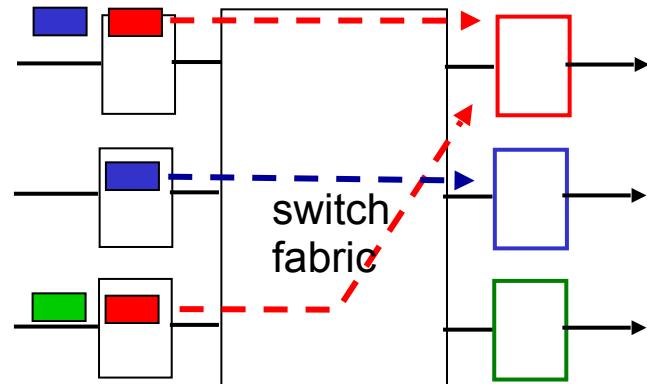
Switching via interconnection network

- scaling, using multiple switching “planes” in parallel:
 - speedup, scaleup via parallelism
- Cisco CRS router:
 - basic unit: 8 switching planes
 - each plane: 3-stage interconnection network
 - up to 100's Tbps switching capacity



Input port queuing

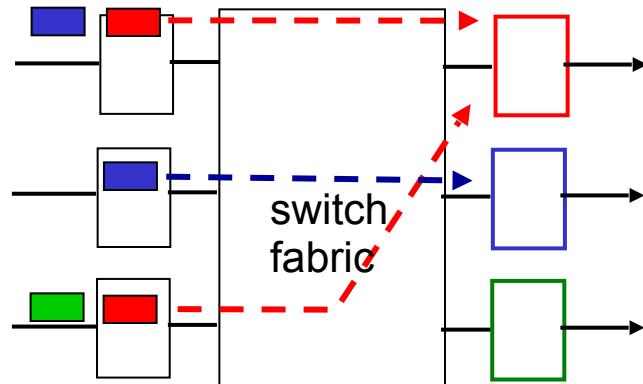
- If switch fabric slower than input ports combined -> queueing may occur at input queues
 - queueing delay and loss due to input buffer overflow!



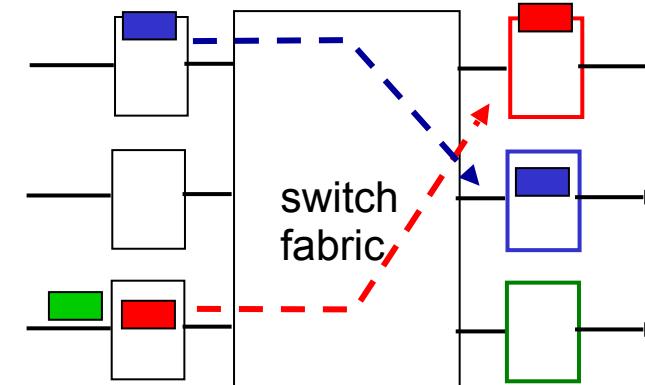
output port contention: only one red datagram can be transferred. lower red packet is *blocked*

Input port queuing

- If switch fabric slower than input ports combined -> queueing may occur at input queues
 - queueing delay and loss due to input buffer overflow!
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward

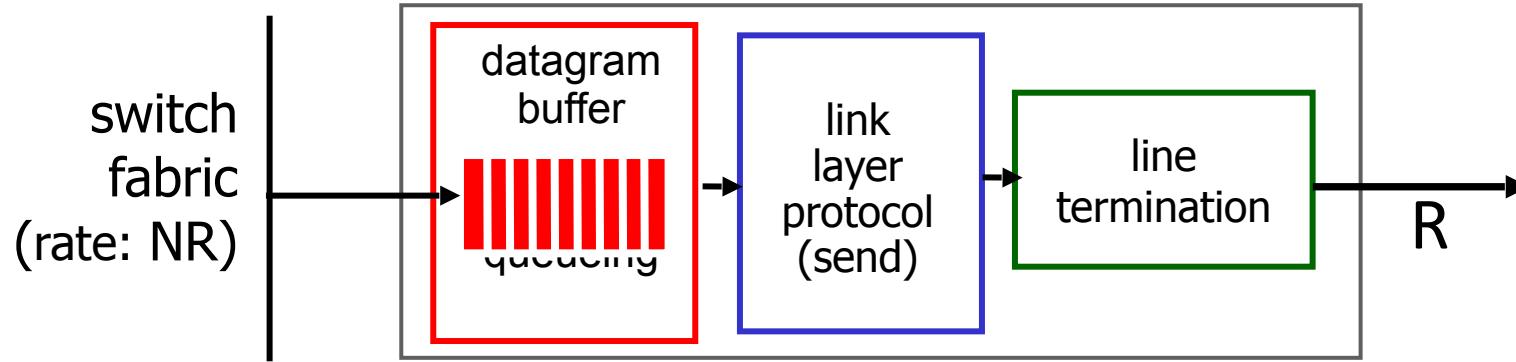


output port contention: only one red datagram can be transferred. lower red packet is *blocked*



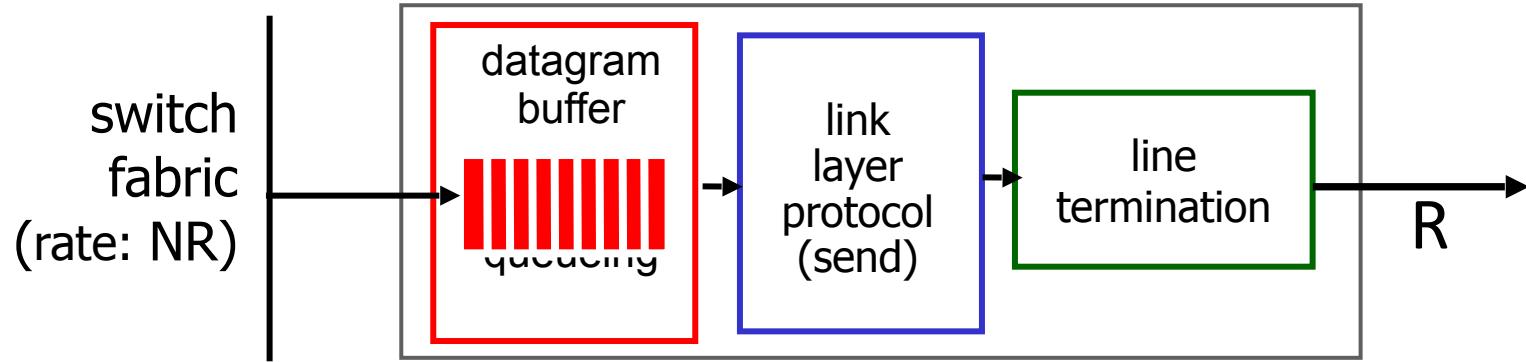
one packet time later: green packet experiences HOL blocking

Output port queuing



- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy*: which datagrams to drop if no free buffers?

Output port queuing

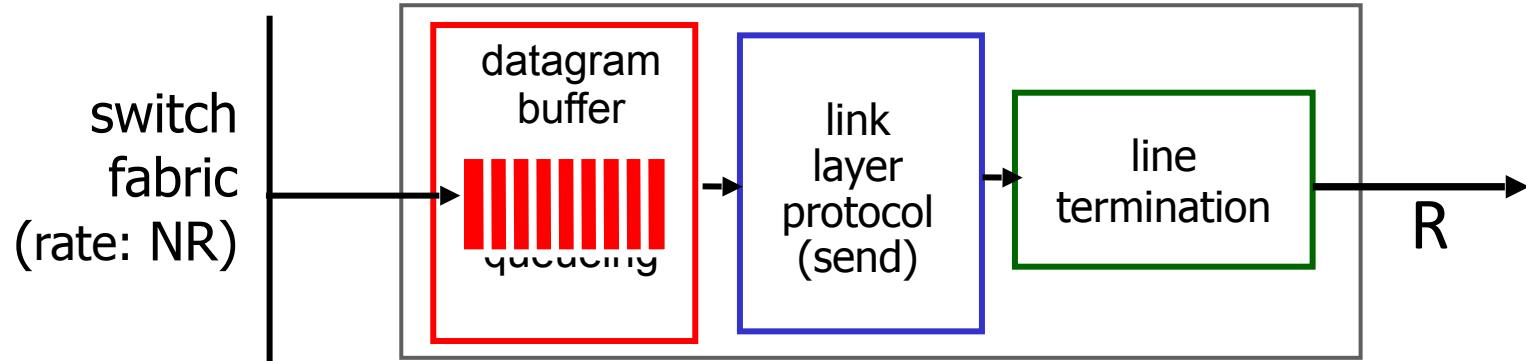


- **Buffering** required when datagrams arrive from fabric faster than link transmission rate. **Drop policy:** which datagrams to drop if no free buffers?



Datagrams can be lost due to congestion, lack of buffers

Output port queuing



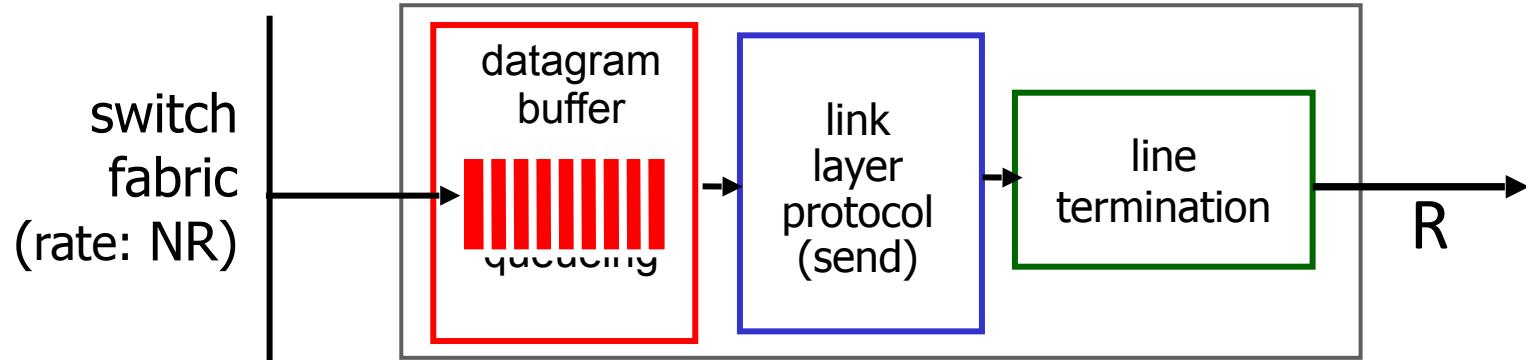
- *Buffering* required when datagrams arrive from fabric faster than link transmission rate. *Drop policy*: which datagrams to drop if no free buffers?



Datagrams can be lost due to congestion, lack of buffers

- *Scheduling discipline* chooses among queued datagrams for transmission

Output port queuing



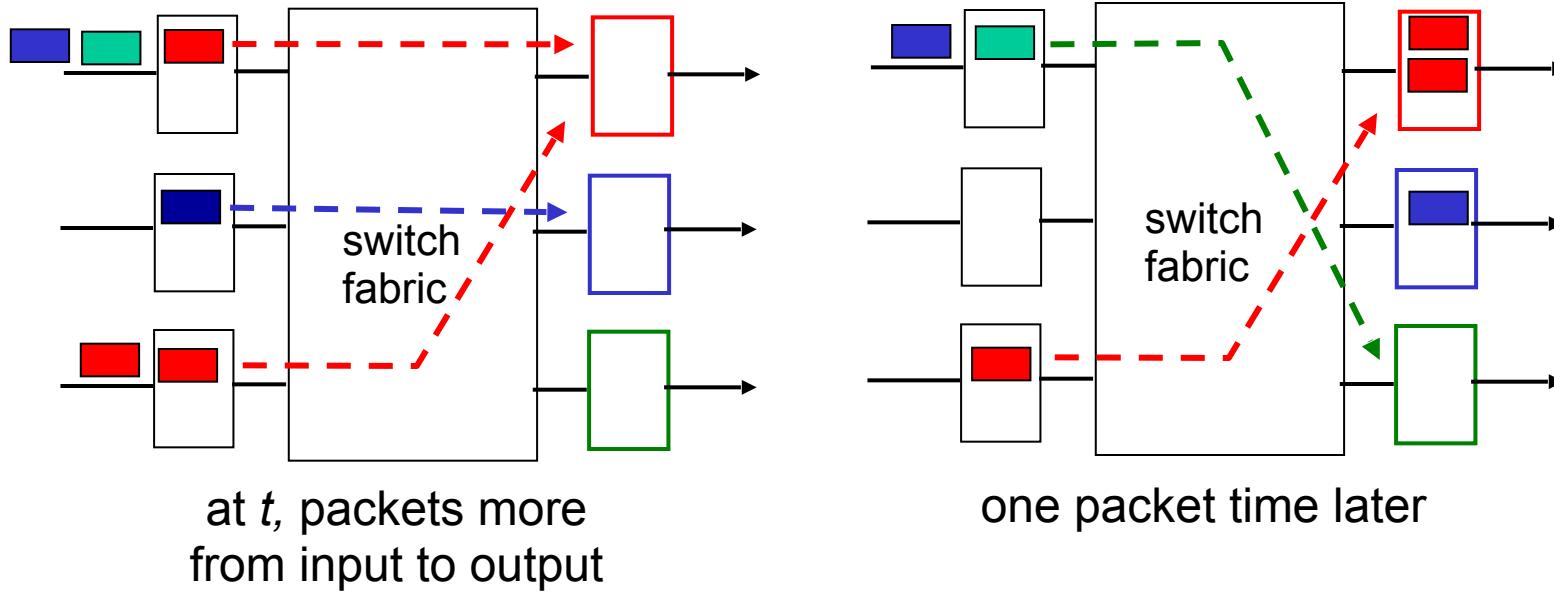
- **Buffering** required when datagrams arrive from fabric faster than link transmission rate. **Drop policy:** which datagrams to drop if no free buffers?

Datagrams can be lost due to congestion, lack of buffers

- **Scheduling discipline** chooses among queued datagrams for transmission

Priority scheduling – who gets best performance, network neutrality

Output port queuing



- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
 - e.g., $C = 10 \text{ Gbps}$ link: 2.5 Gbit buffer

How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
 - e.g., $C = 10 \text{ Gbps}$ link: 2.5 Gbit buffer
- more recent recommendation: with N flows, buffering equal to

$$\frac{\text{RTT} \cdot C}{\sqrt{N}}$$

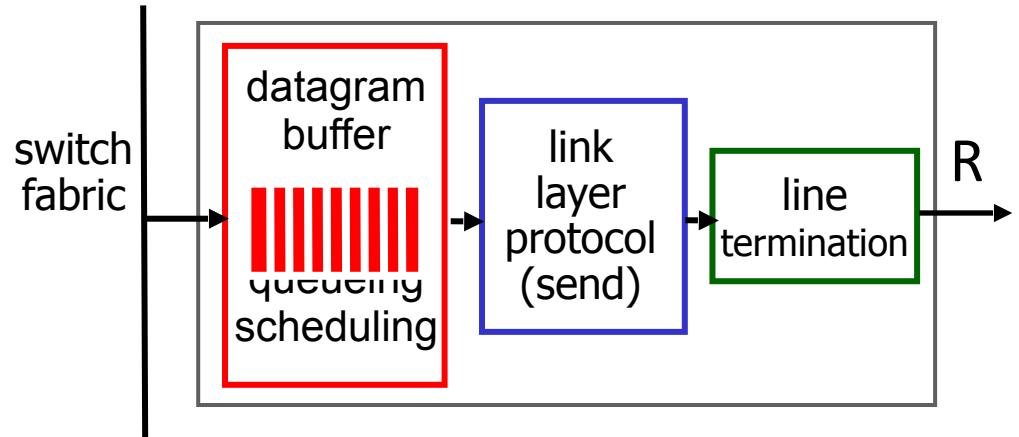
How much buffering?

- RFC 3439 rule of thumb: average buffering equal to “typical” RTT (say 250 msec) times link capacity C
 - e.g., C = 10 Gbps link: 2.5 Gbit buffer
- more recent recommendation: with N flows, buffering equal to

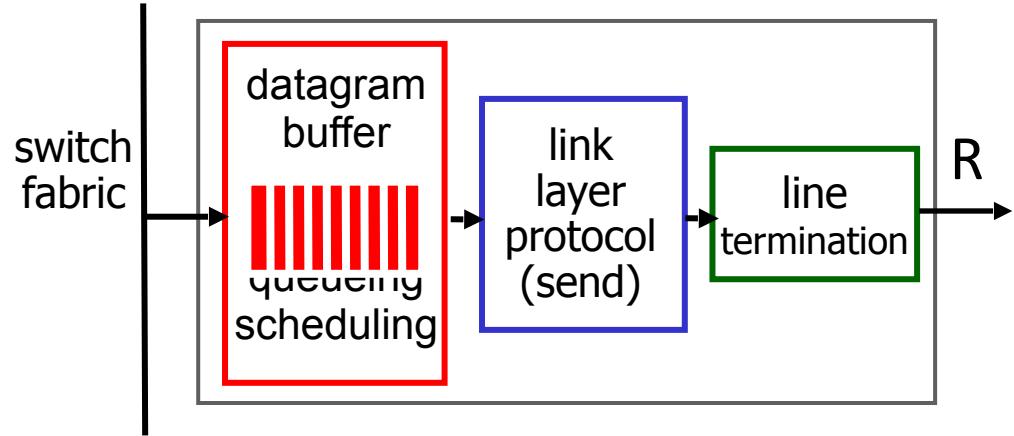
$$\frac{\text{RTT} \cdot C}{\sqrt{N}}$$

- but *too* much buffering can increase delays (particularly in home routers)
 - long RTTs: poor performance for realtime apps, sluggish TCP response
 - recall delay-based congestion control: “keep bottleneck link just full enough (busy) but no fuller”

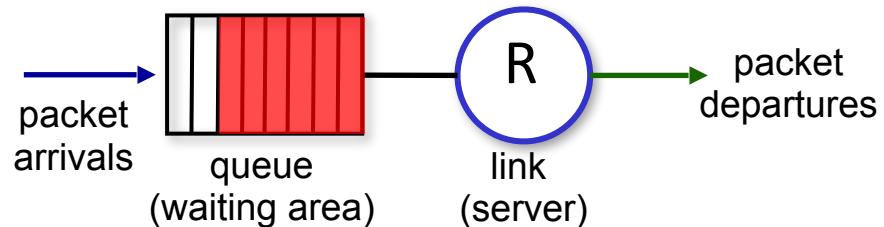
Buffer Management



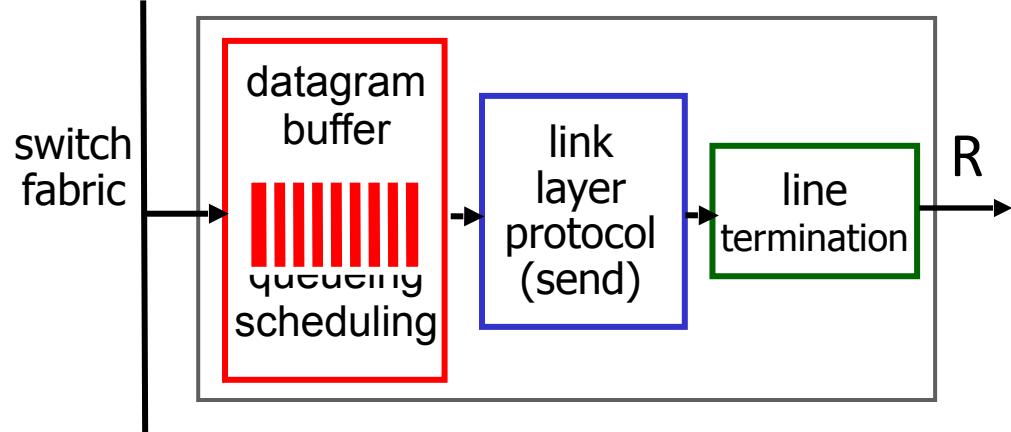
Buffer Management



Abstraction: queue



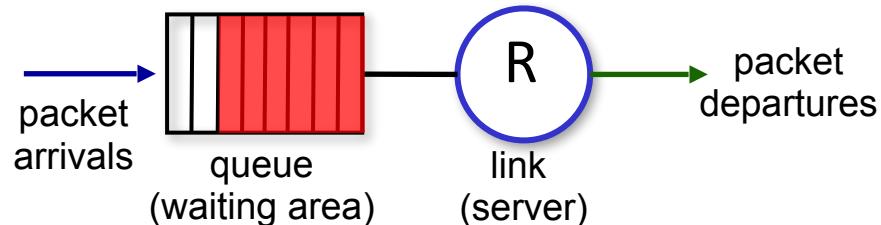
Buffer Management



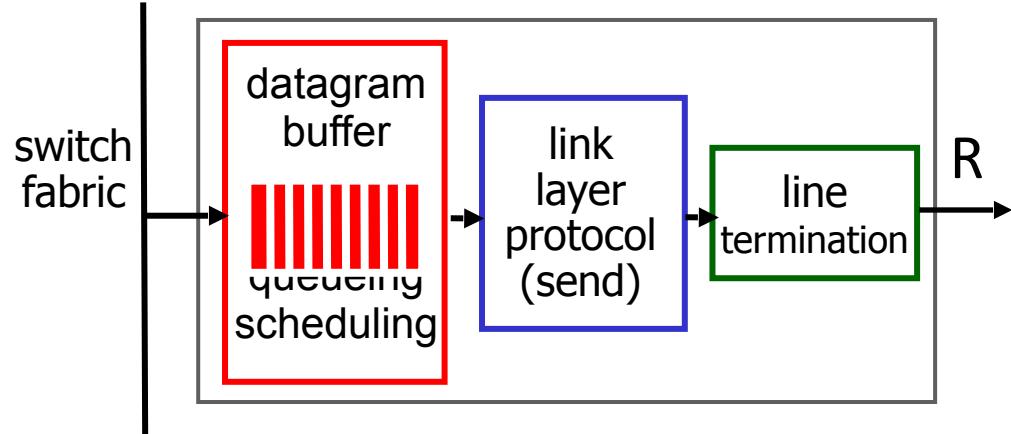
buffer management:

- **drop:** which packet to add, drop when buffers are full
 - **tail drop:** drop arriving packet
 - **priority:** drop/remove on priority basis

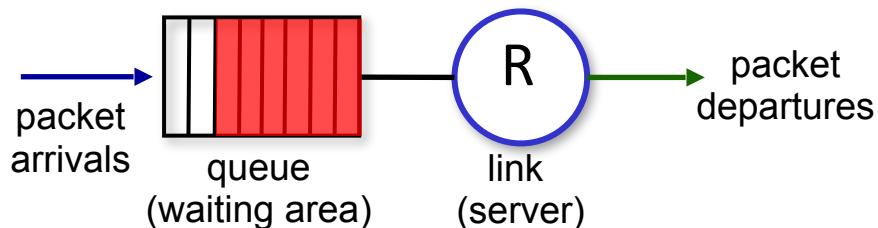
Abstraction: queue



Buffer Management



Abstraction: queue



buffer management:

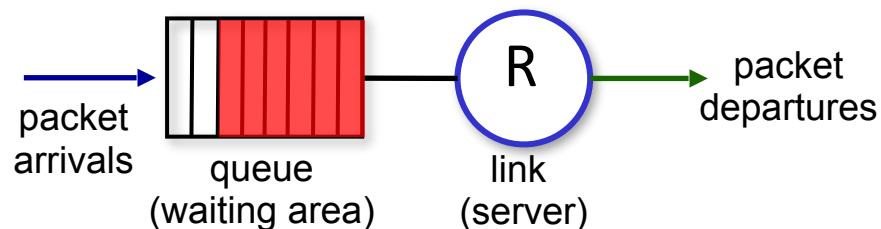
- **drop:** which packet to add, drop when buffers are full
 - **tail drop:** drop arriving packet
 - **priority:** drop/remove on priority basis
- **marking:** which packets to mark to signal congestion (ECN, RED)

Packet Scheduling: FCFS

packet scheduling: deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

Abstraction: queue



Packet Scheduling: FCFS

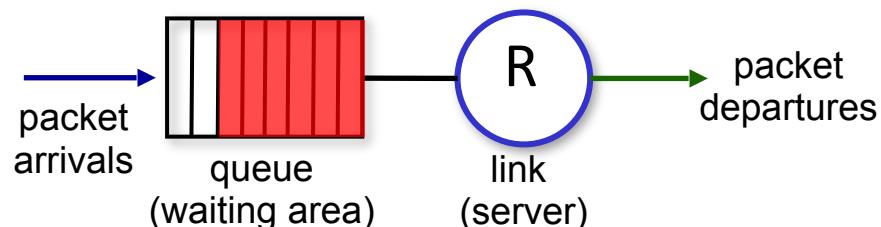
packet scheduling: deciding which packet to send next on link

- first come, first served
- priority
- round robin
- weighted fair queueing

FCFS: packets transmitted in order of arrival to output port

- also known as: First-in-first-out (FIFO)
- real world examples?

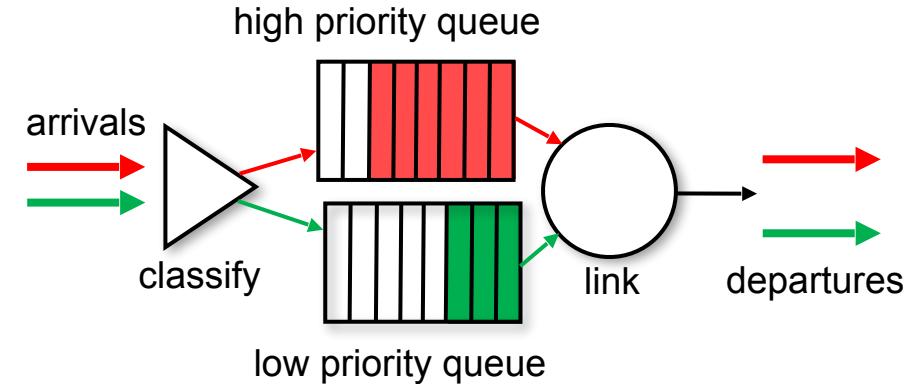
Abstraction: queue



Scheduling policies: priority

Priority scheduling:

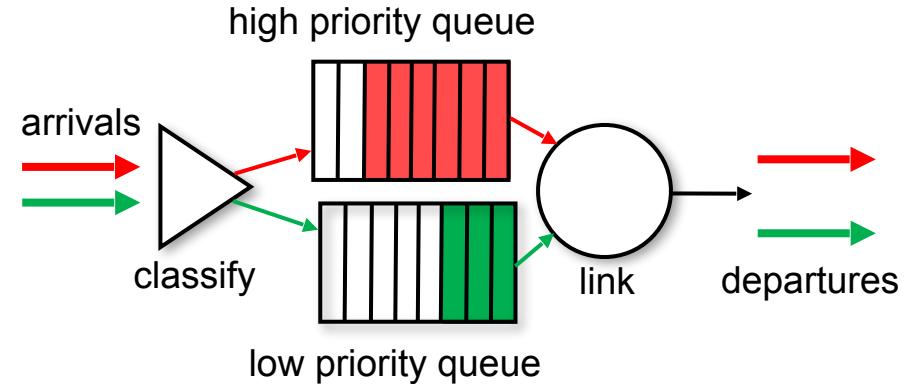
- arriving traffic classified, queued by class
 - any header fields can be used for classification



Scheduling policies: priority

Priority scheduling:

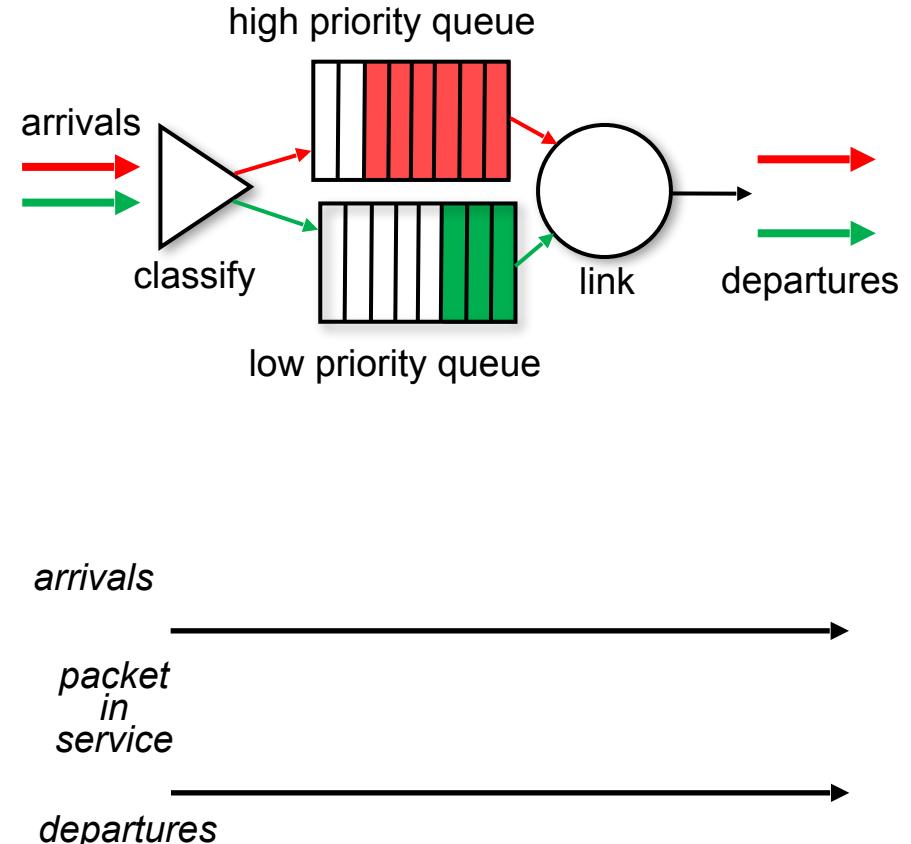
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

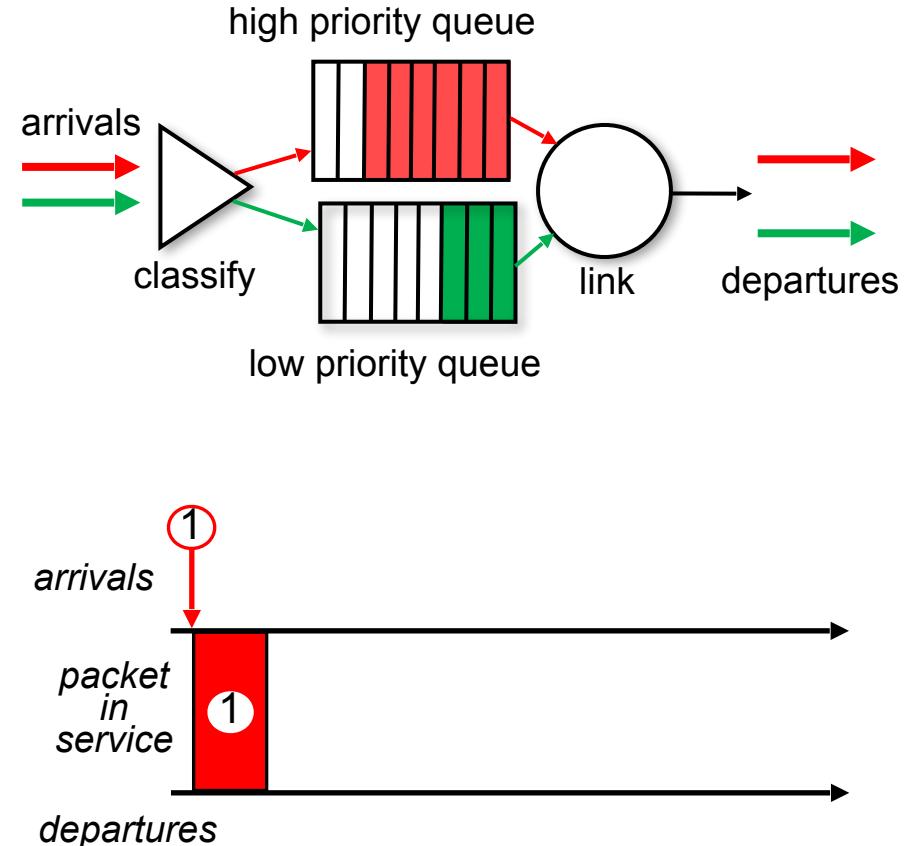
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

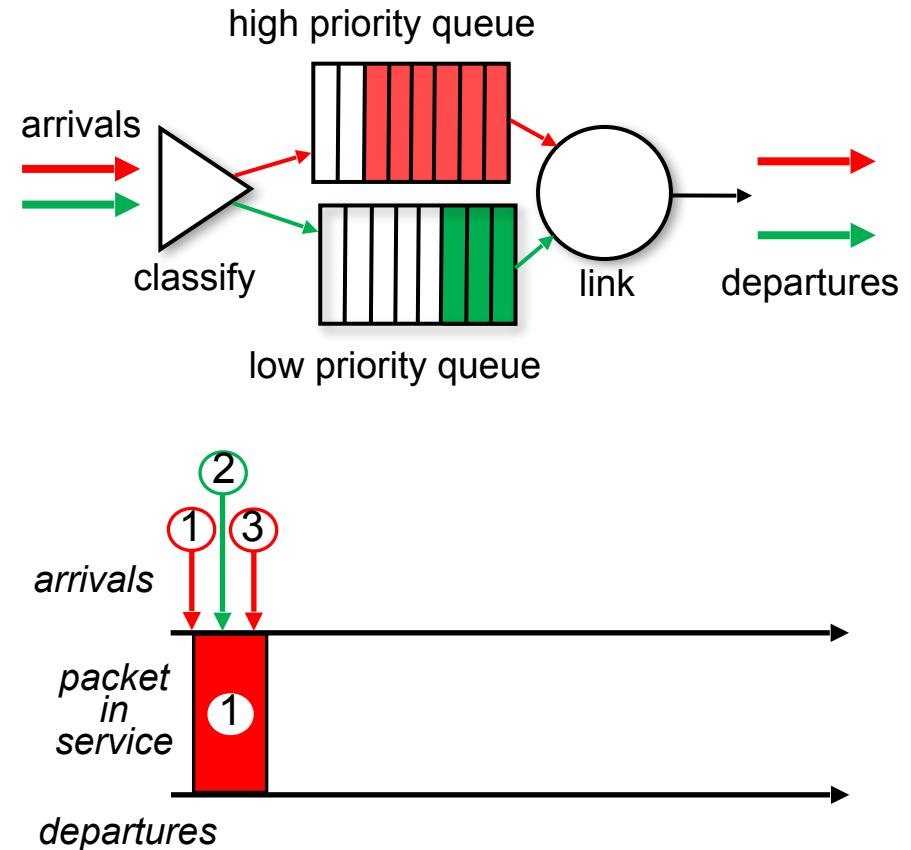
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

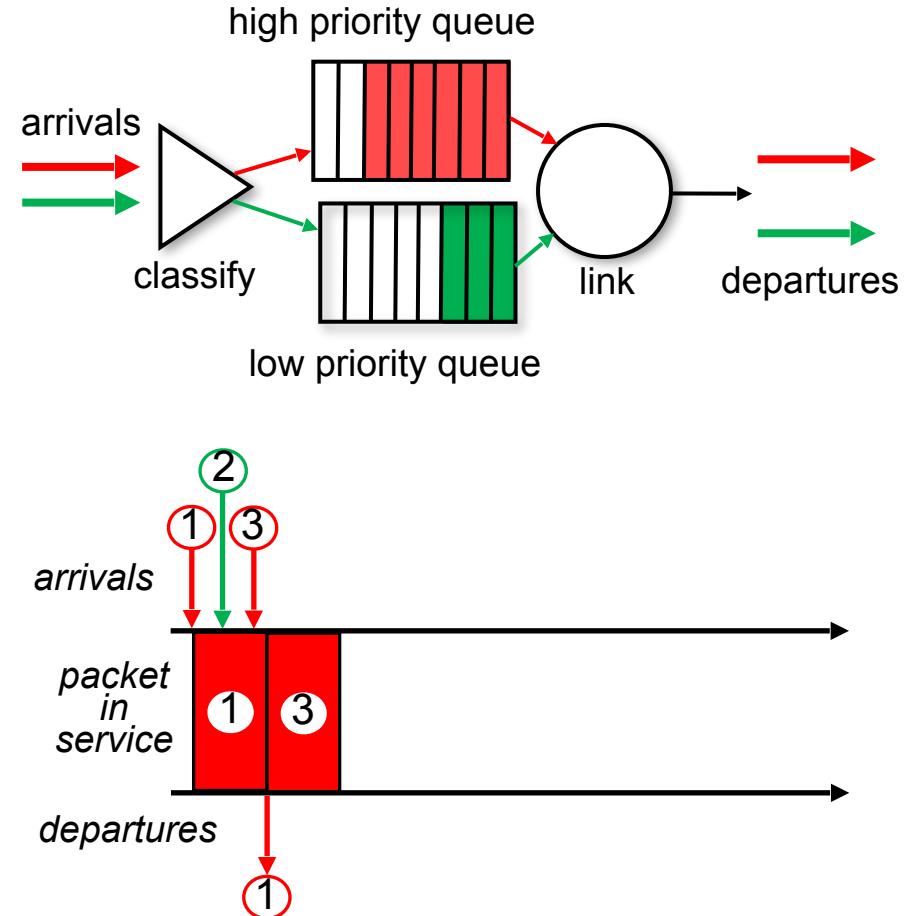
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

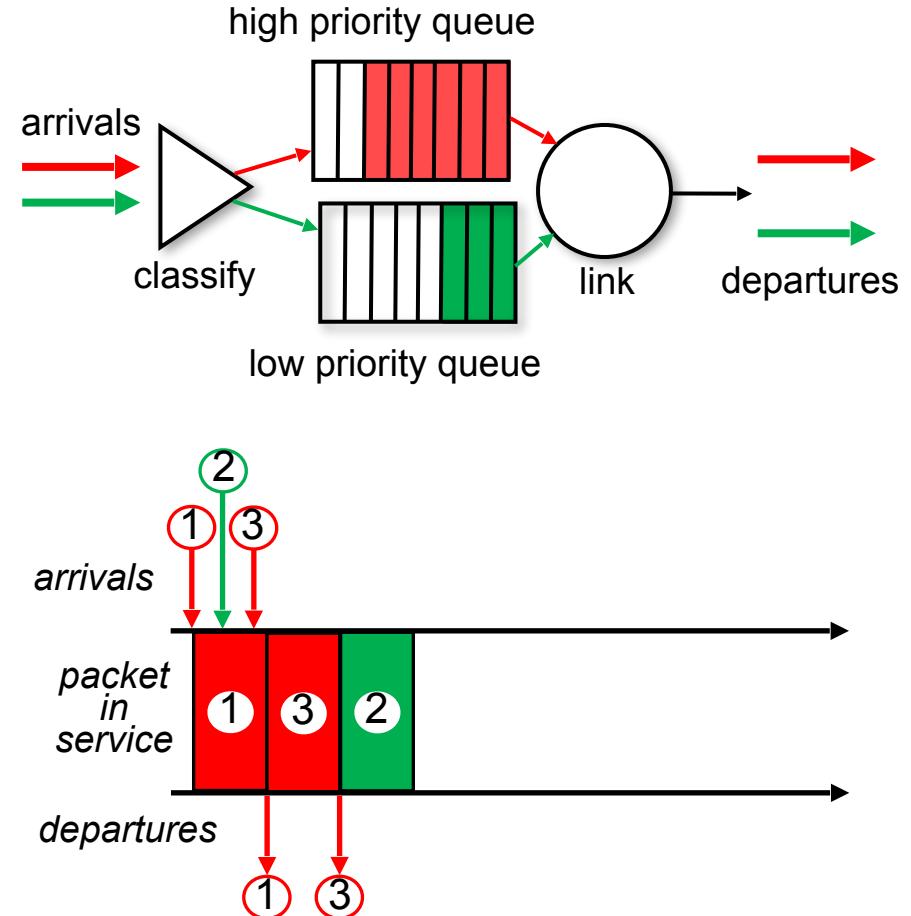
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

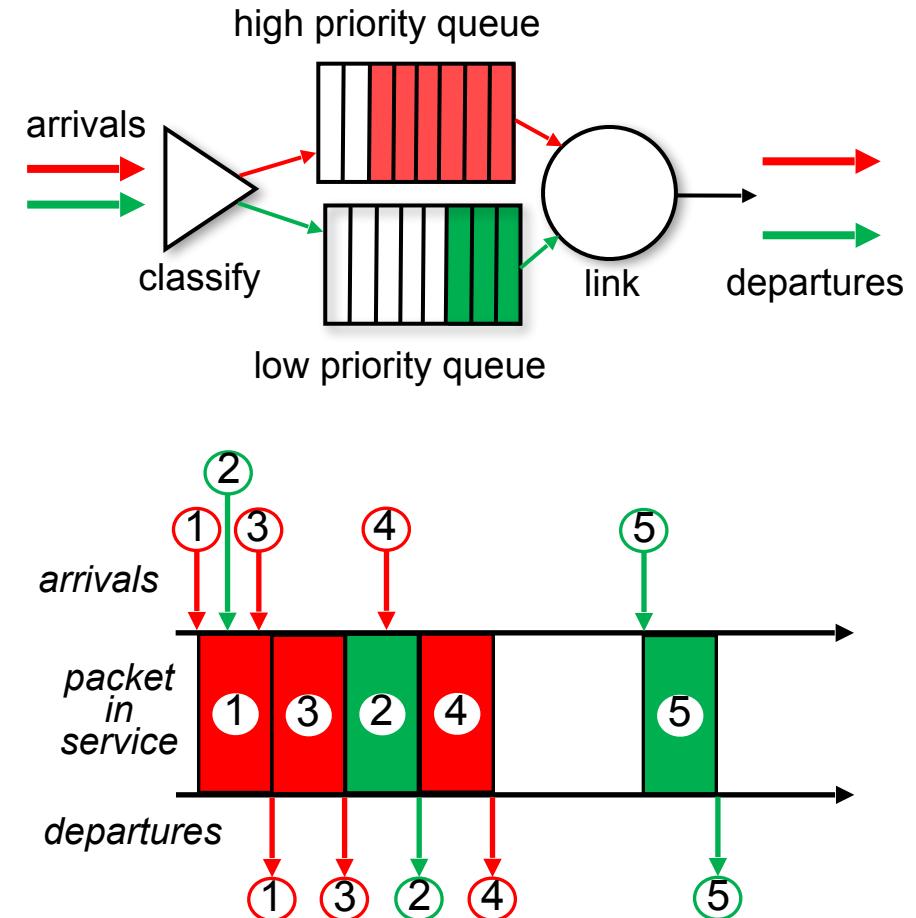
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: priority

Priority scheduling:

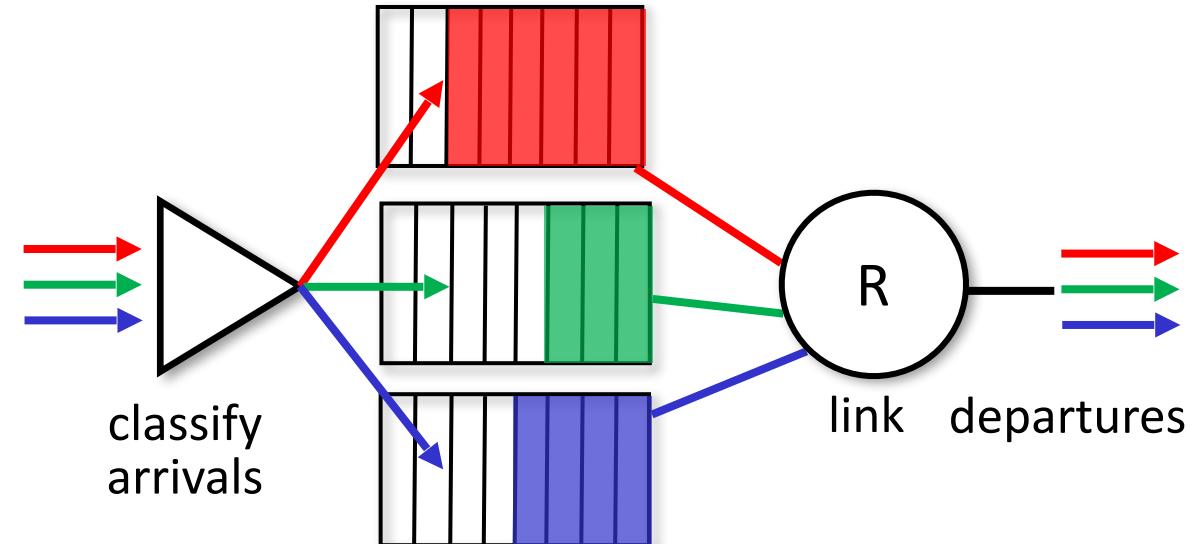
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- send packet from highest priority queue that has buffered packets
 - FCFS within priority class



Scheduling policies: round robin

Round Robin (RR) scheduling:

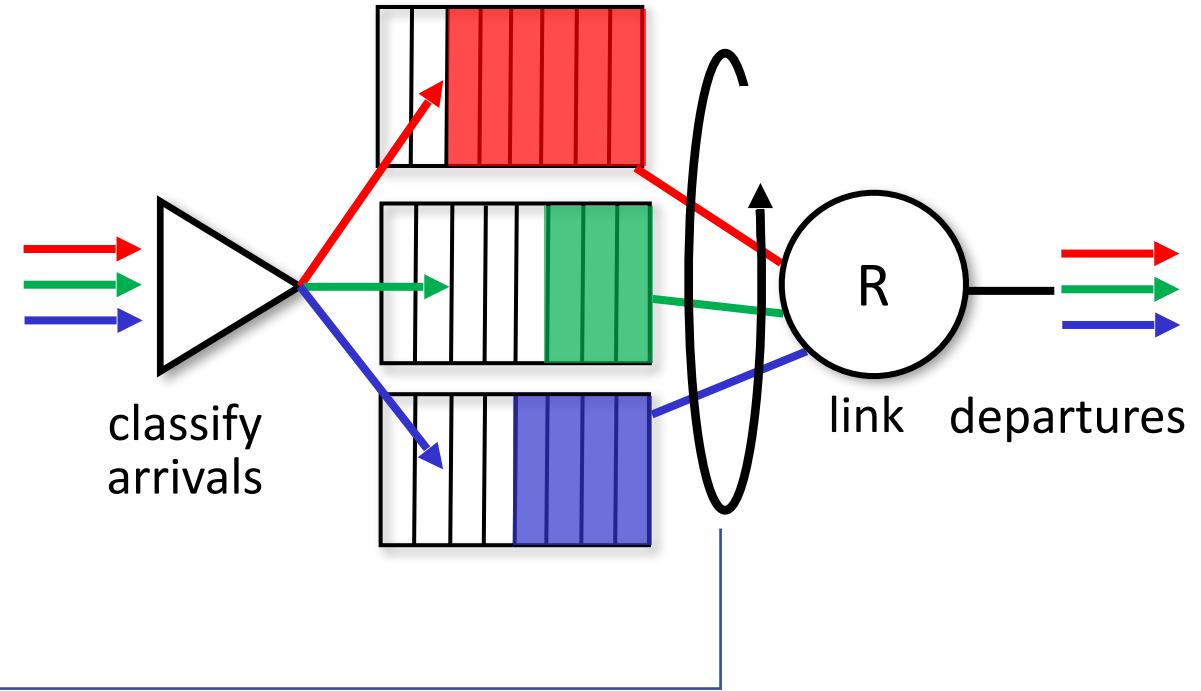
- arriving traffic classified, queued by class
 - any header fields can be used for classification



Scheduling policies: round robin

Round Robin (RR) scheduling:

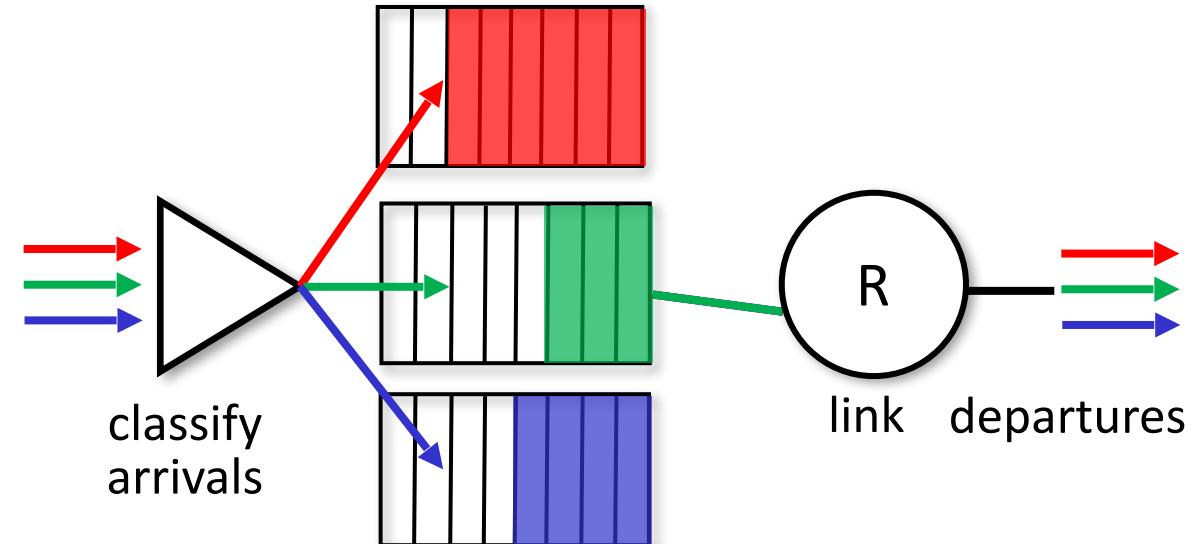
- arriving traffic classified, queued by class
 - any header fields can be used for classification
- server cyclically, repeatedly scans class queues, sending one complete packet from each class (if available) in turn



Scheduling policies: weighted fair queueing

Weighted Fair Queueing (WFQ):

- generalized Round Robin

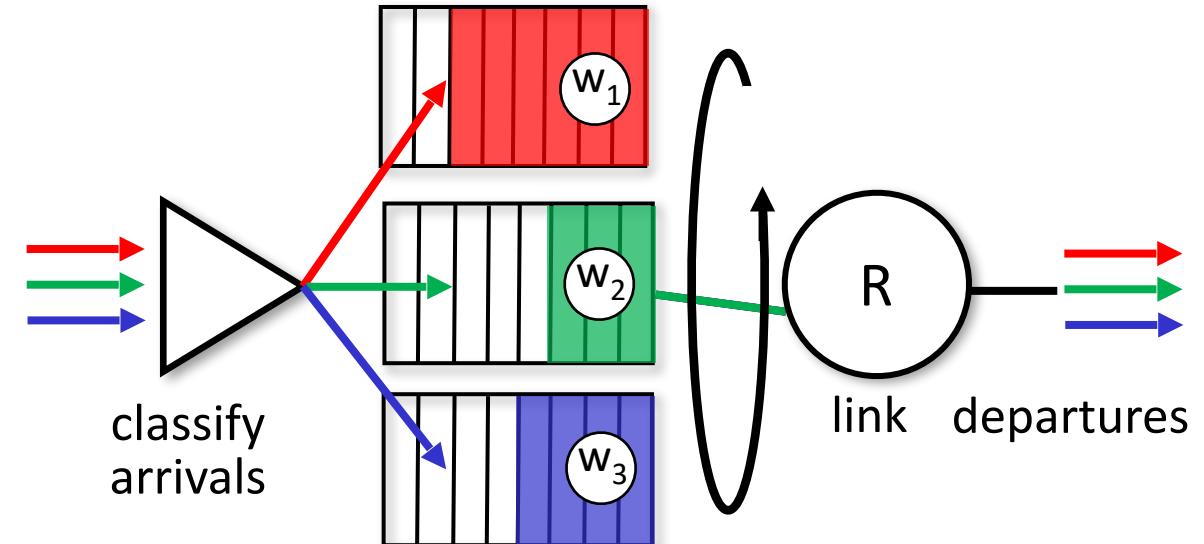


Scheduling policies: weighted fair queueing

Weighted Fair Queuing (WFQ):

- generalized Round Robin
- each class, i , has weight, w_i ,
and gets weighted amount of
service in each cycle:

$$\frac{w_i}{\sum_j w_j}$$



Scheduling policies: weighted fair queueing

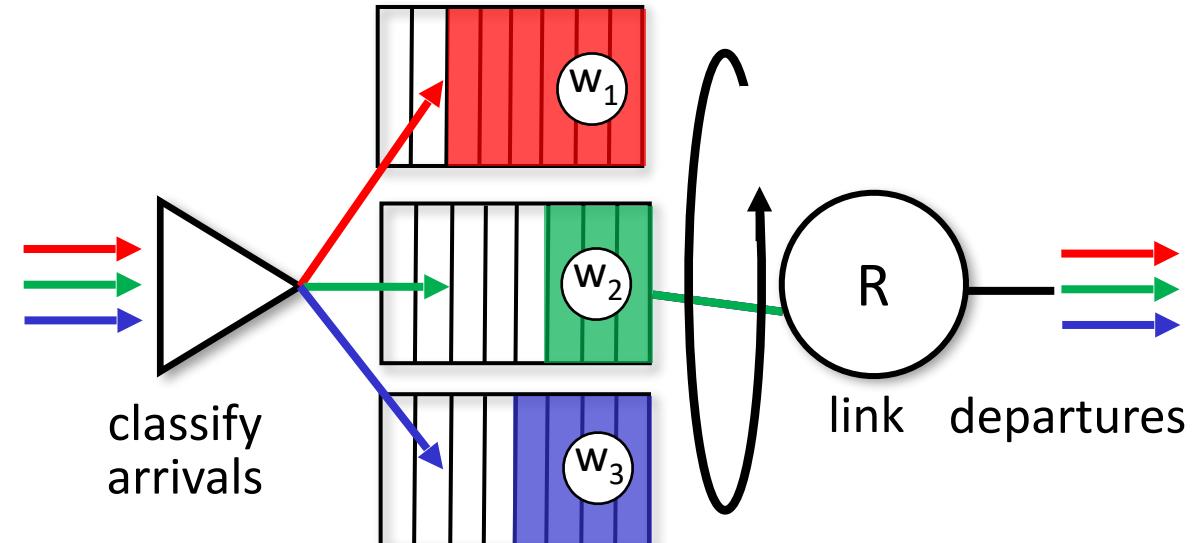
Weighted Fair Queueing (WFQ):

- generalized Round Robin
- each class, i , has weight, w_i ,
and gets weighted amount of
service in each cycle:

$$w_i$$

$$\sum_j w_j$$

- minimum bandwidth
guarantee (per-traffic-class)

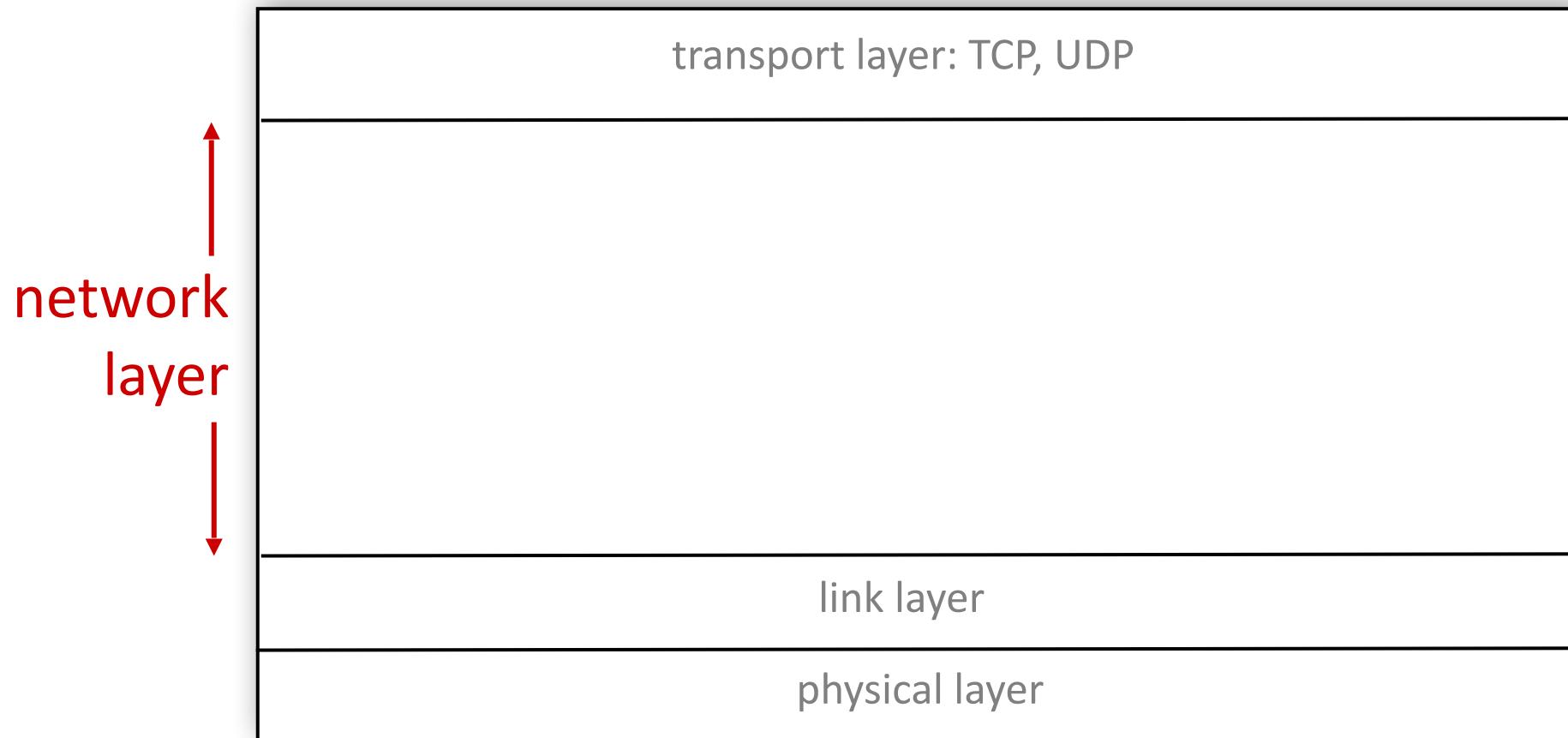


Chapter 4: roadmap

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - **datagram format**
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

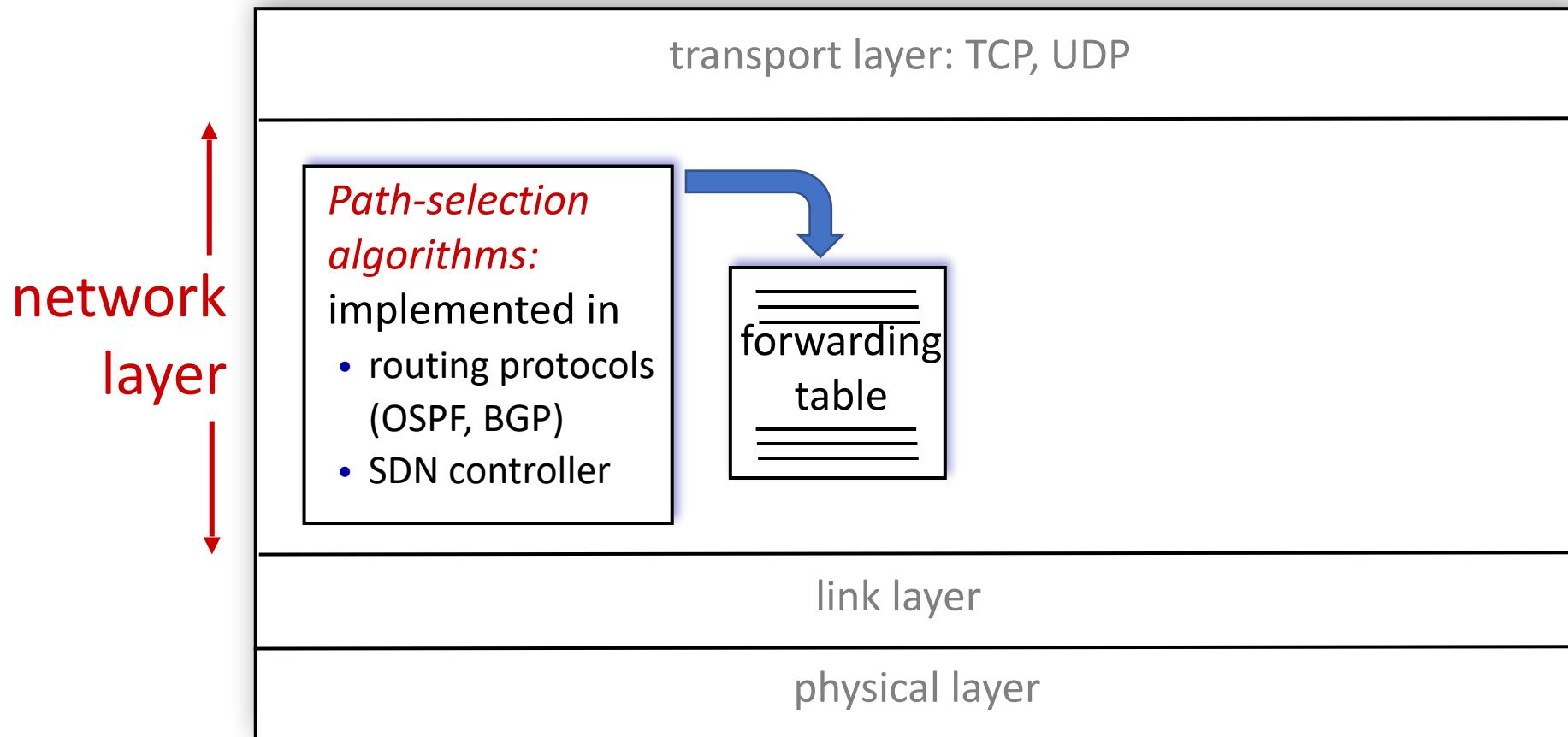
Network Layer: Internet

host, router network layer functions:



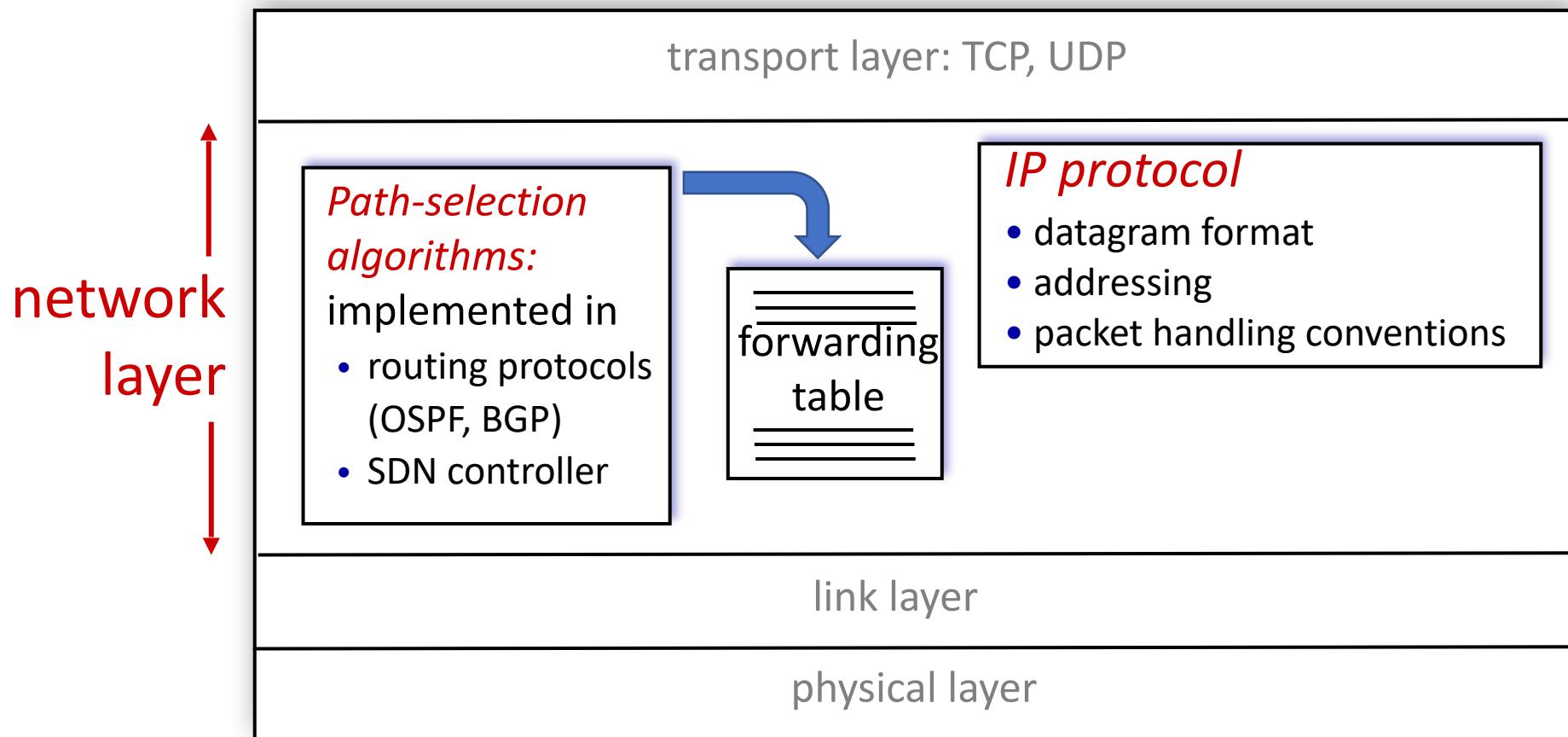
Network Layer: Internet

host, router network layer functions:



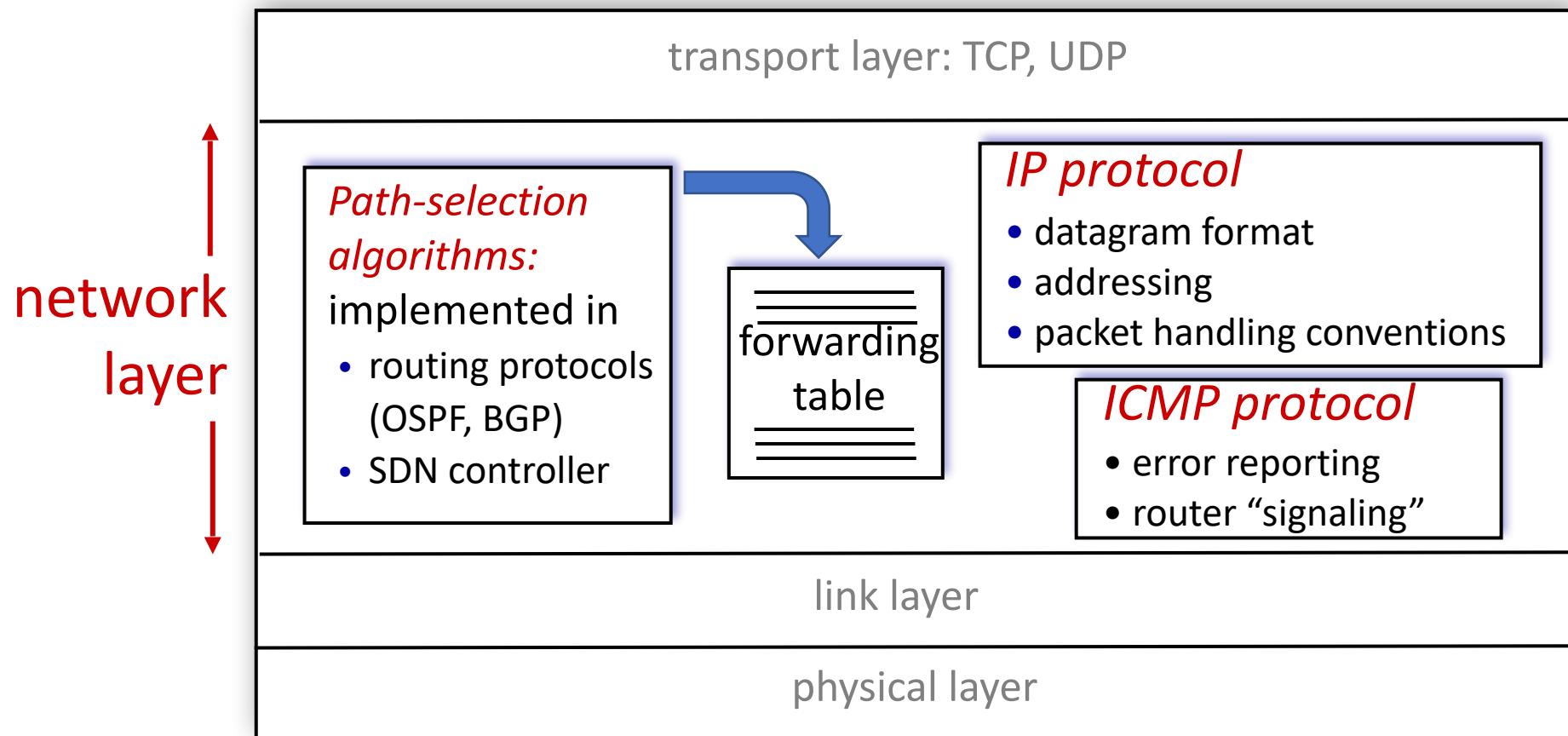
Network Layer: Internet

host, router network layer functions:

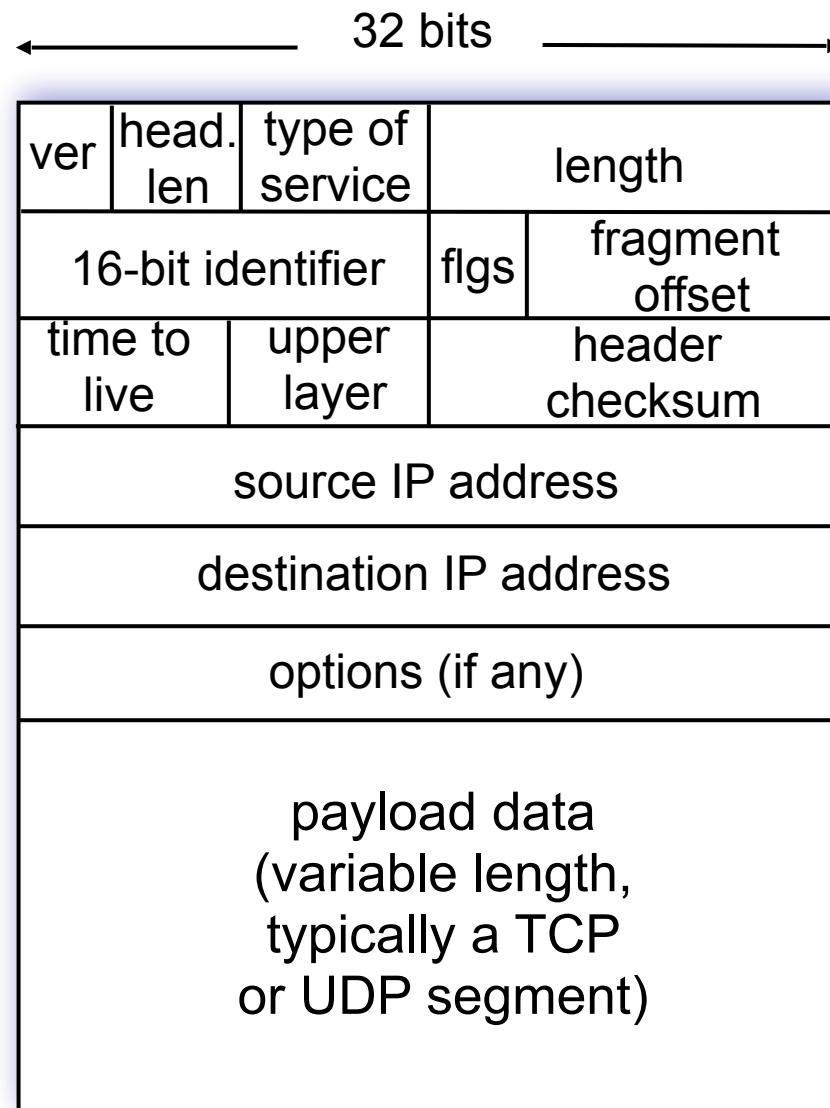


Network Layer: Internet

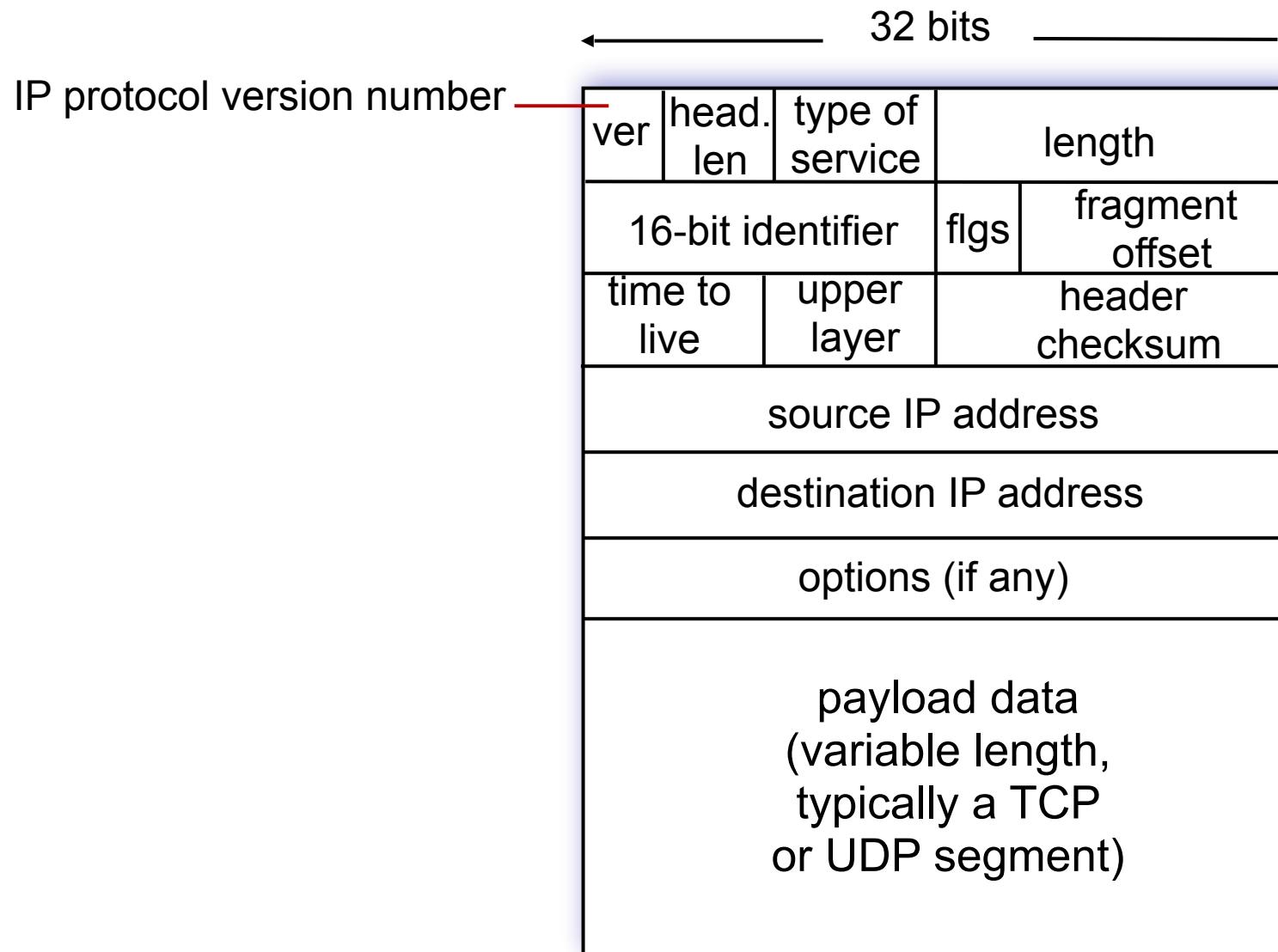
host, router network layer functions:



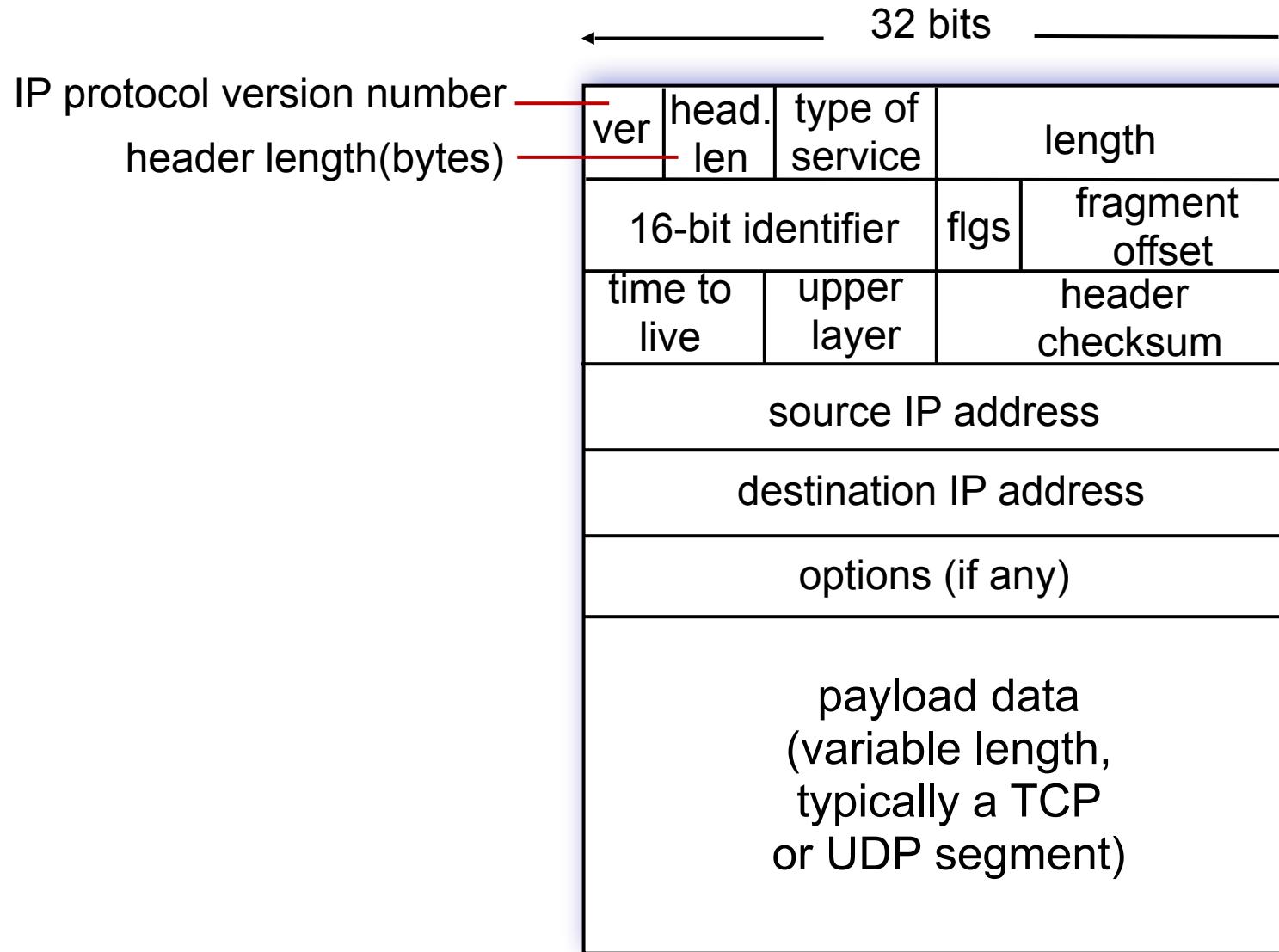
IP Datagram format



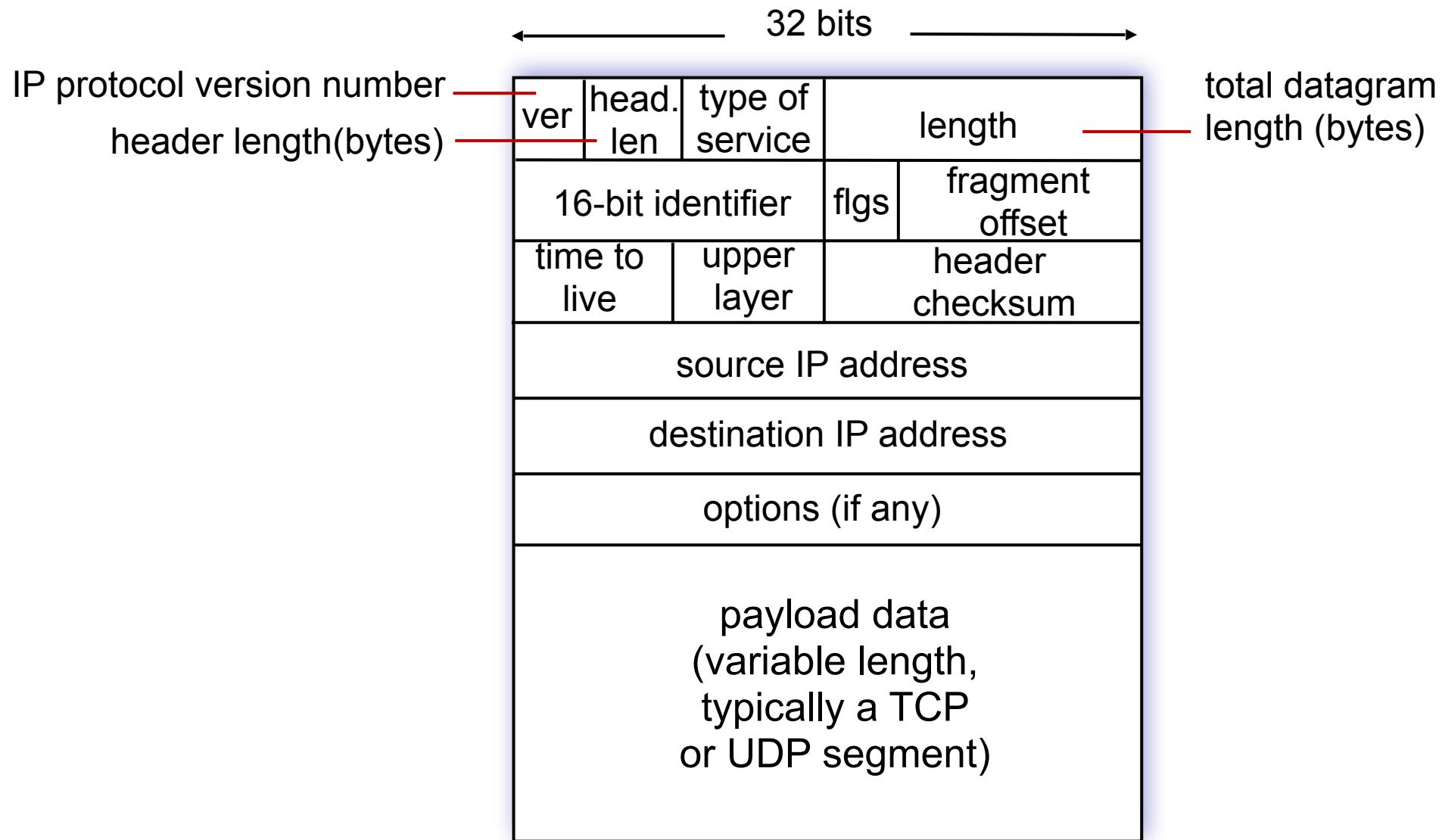
IP Datagram format



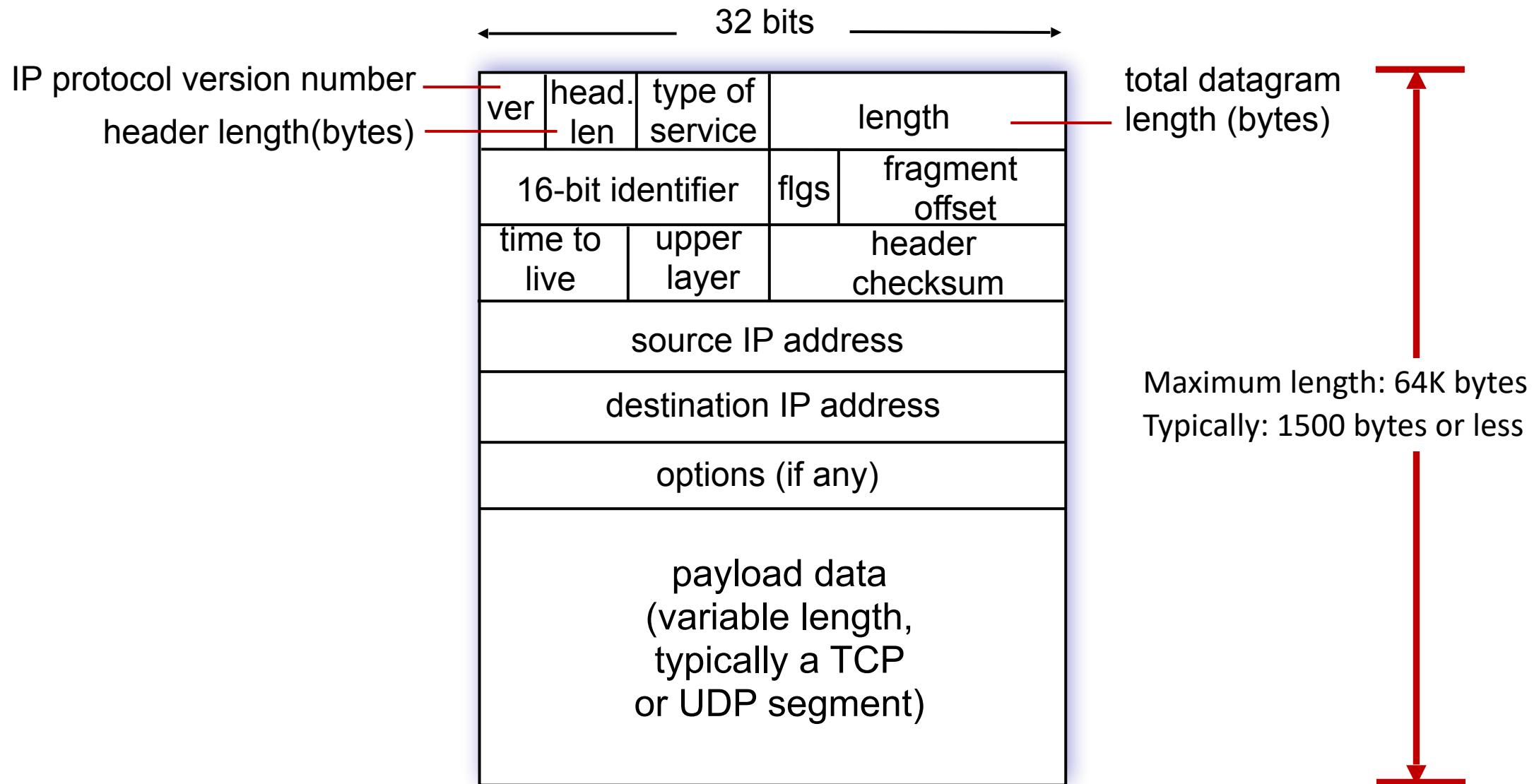
IP Datagram format



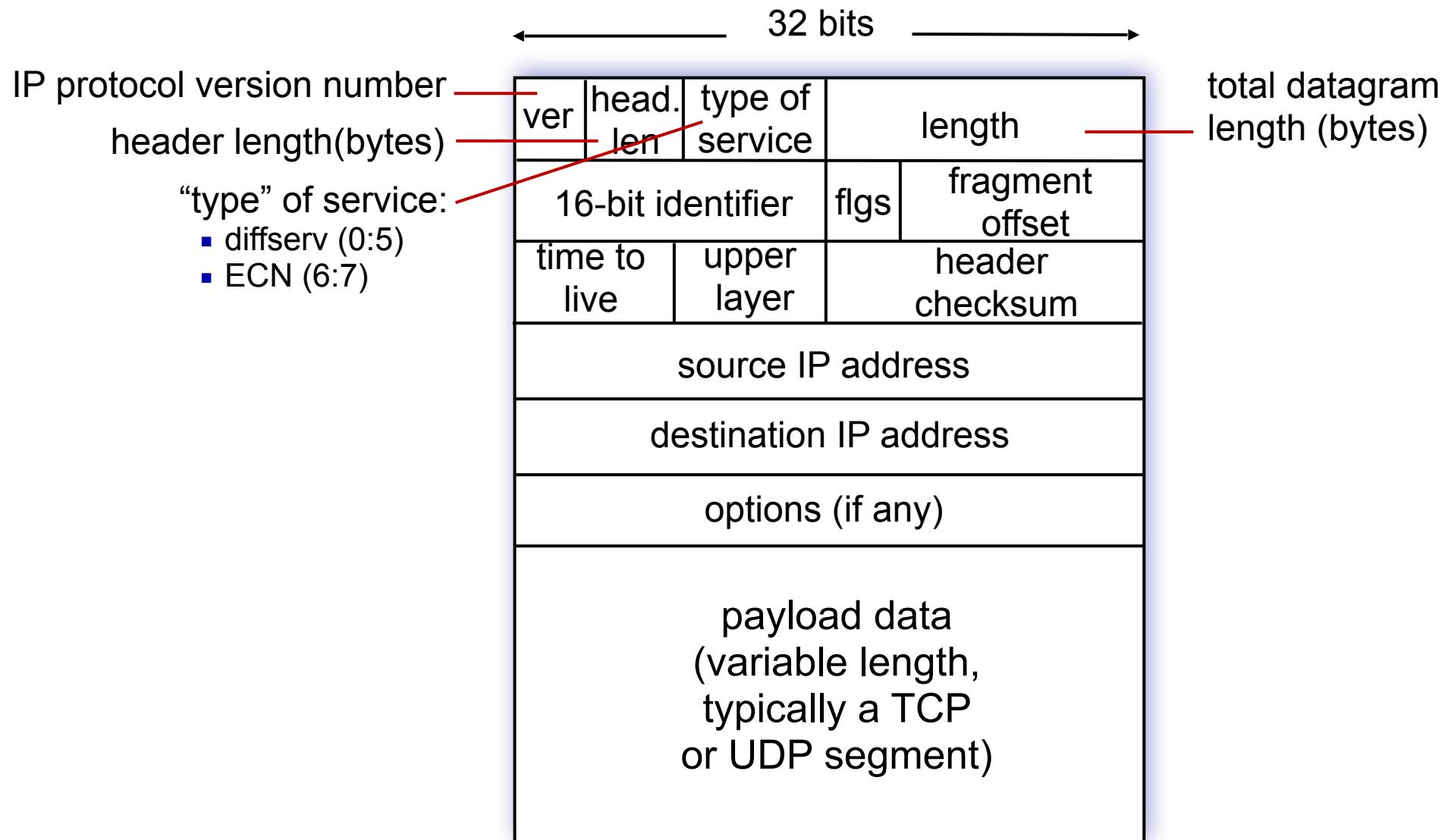
IP Datagram format



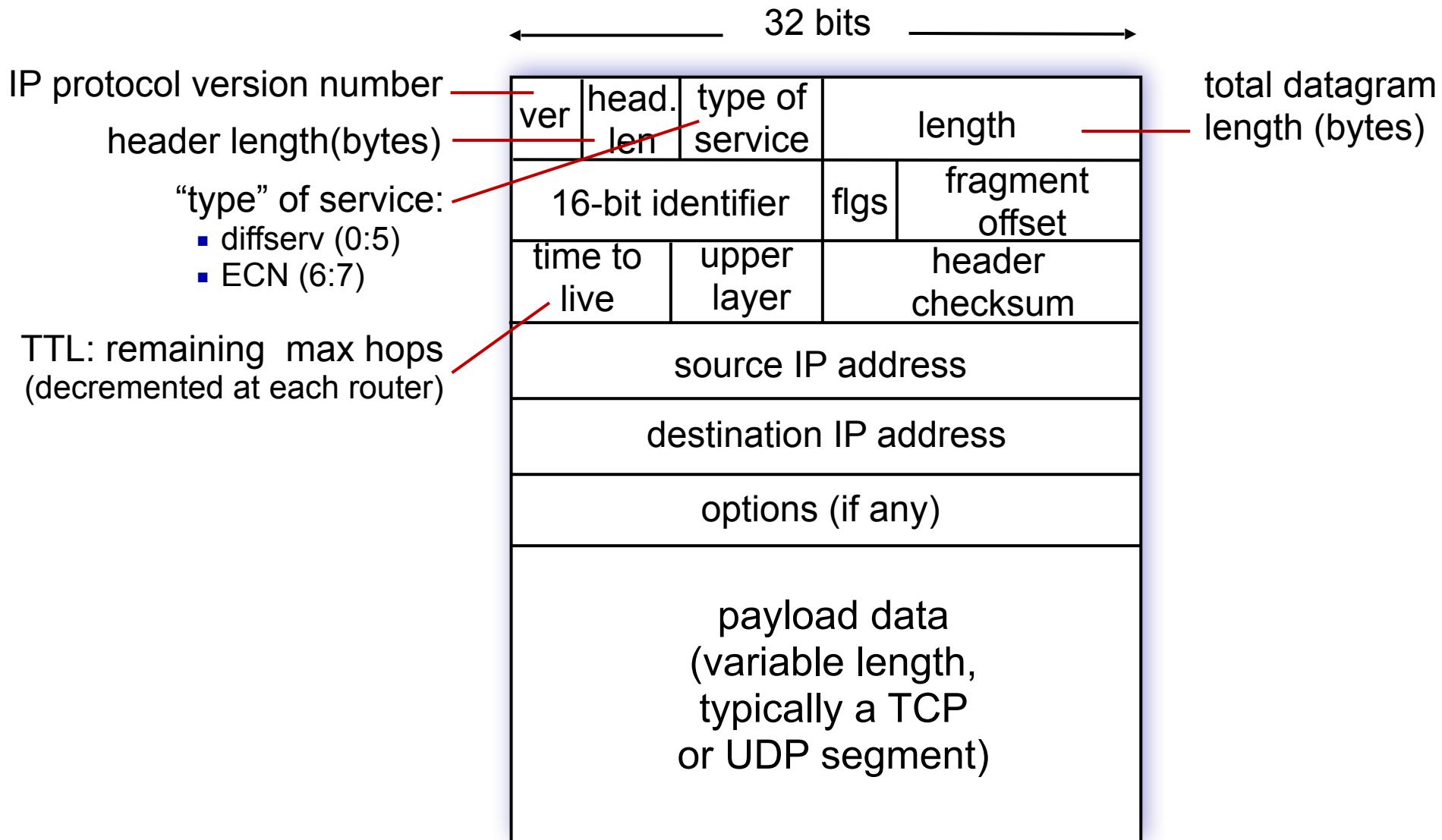
IP Datagram format



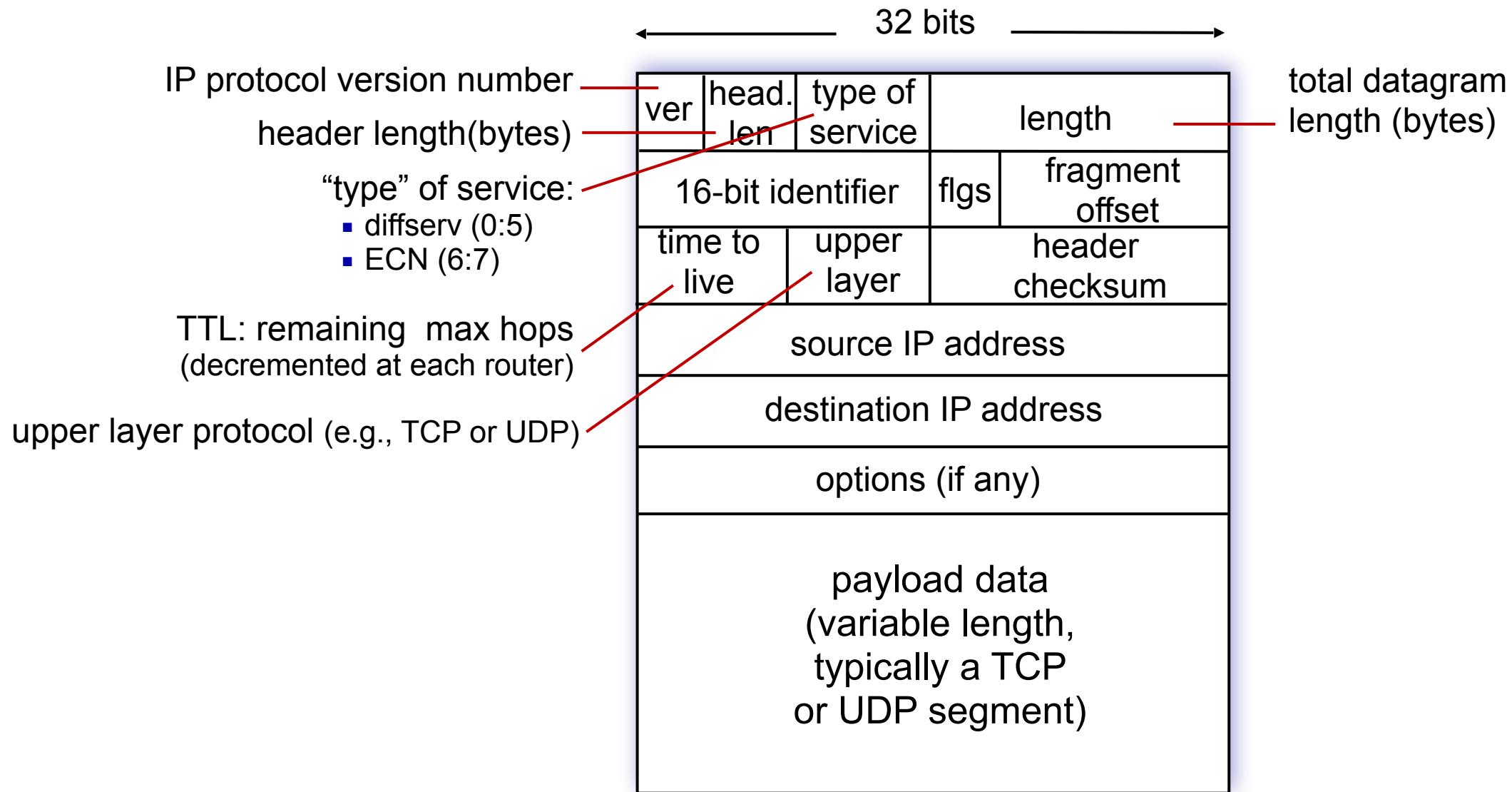
IP Datagram format



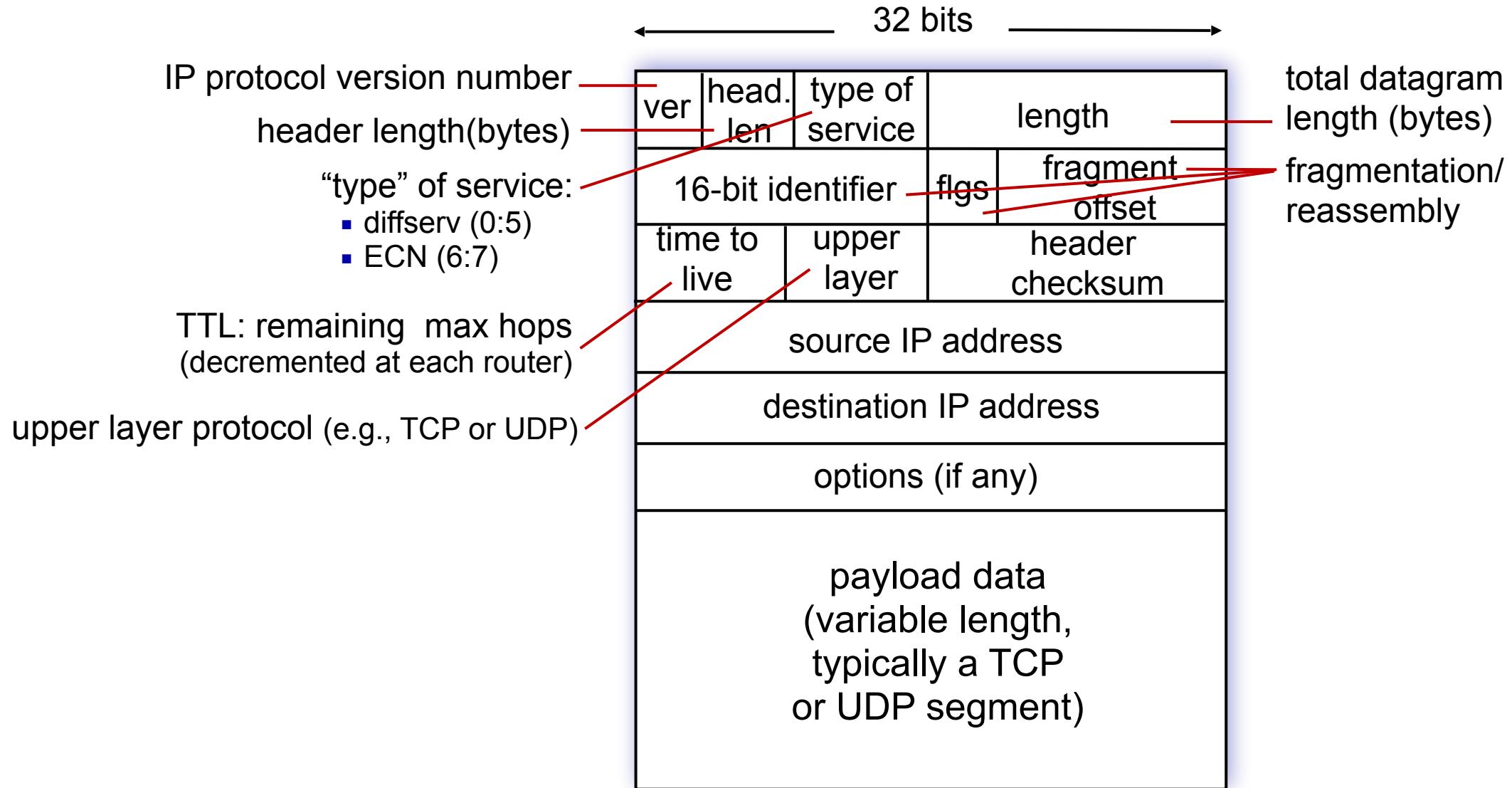
IP Datagram format



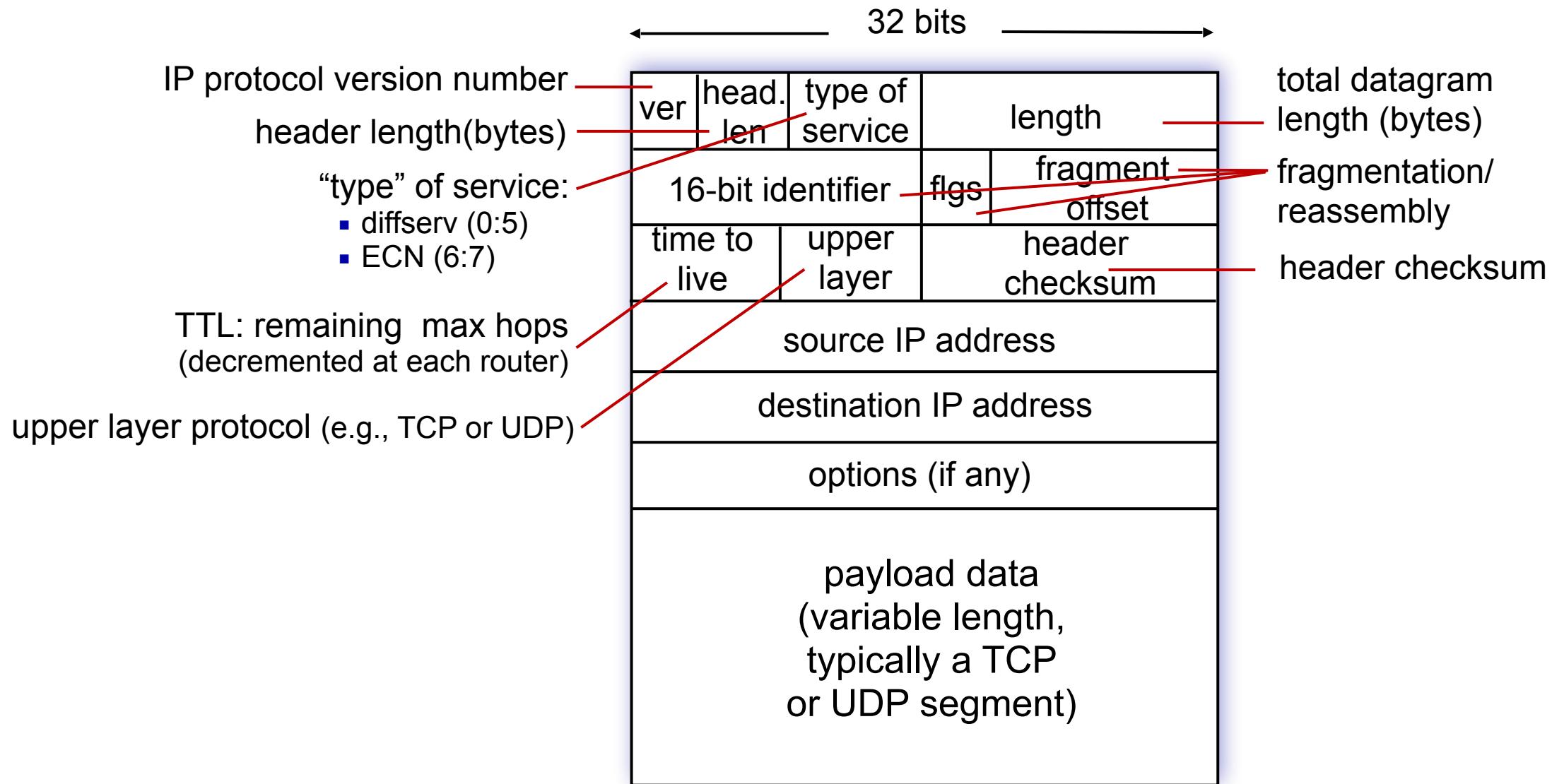
IP Datagram format



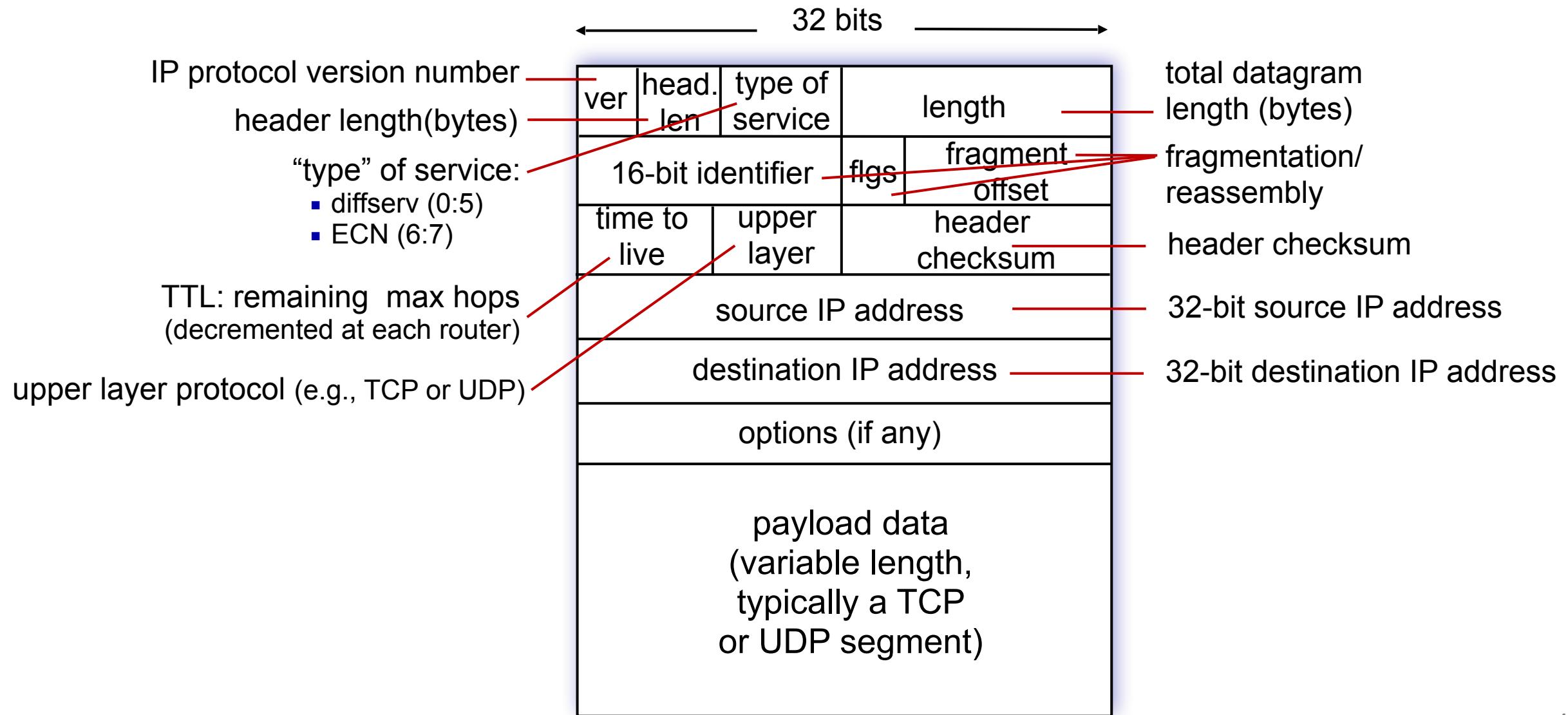
IP Datagram format



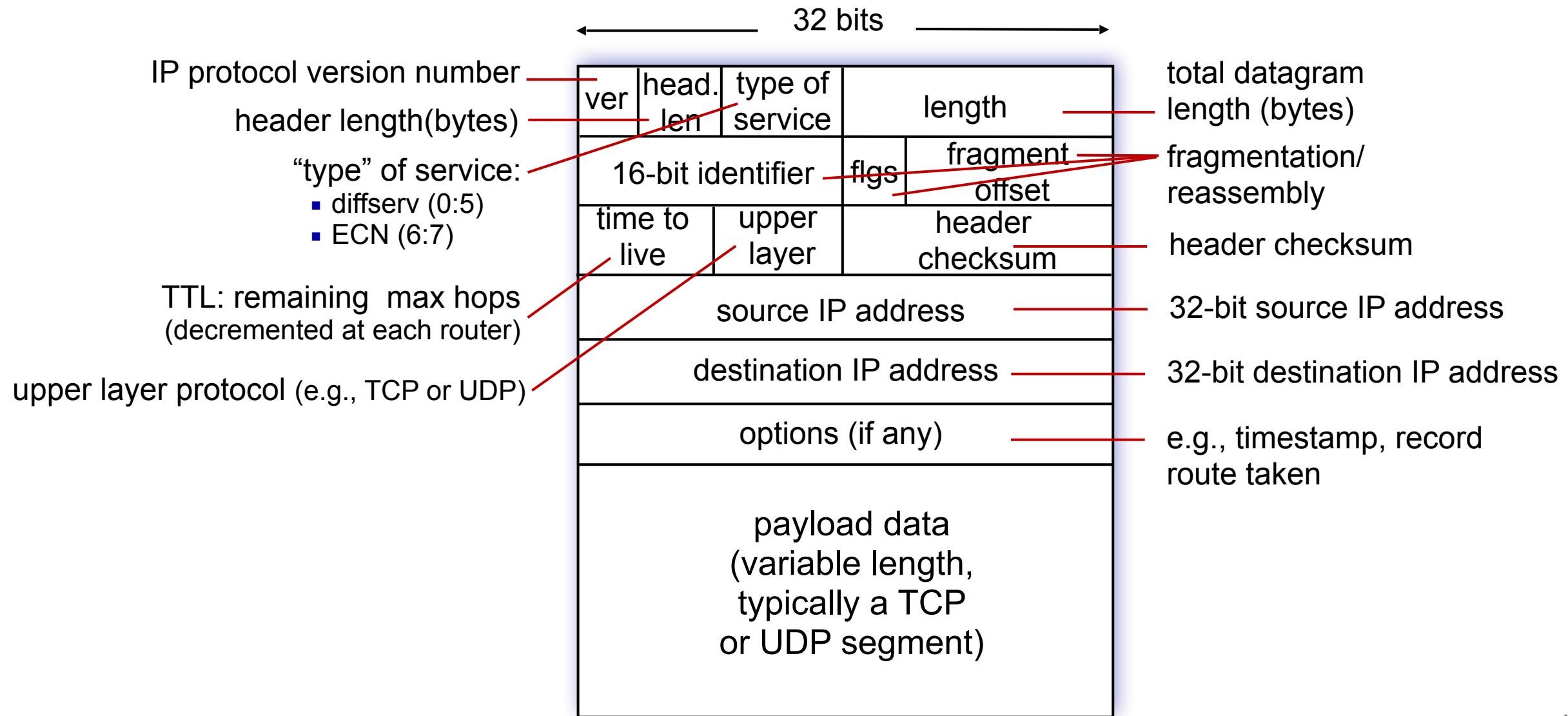
IP Datagram format



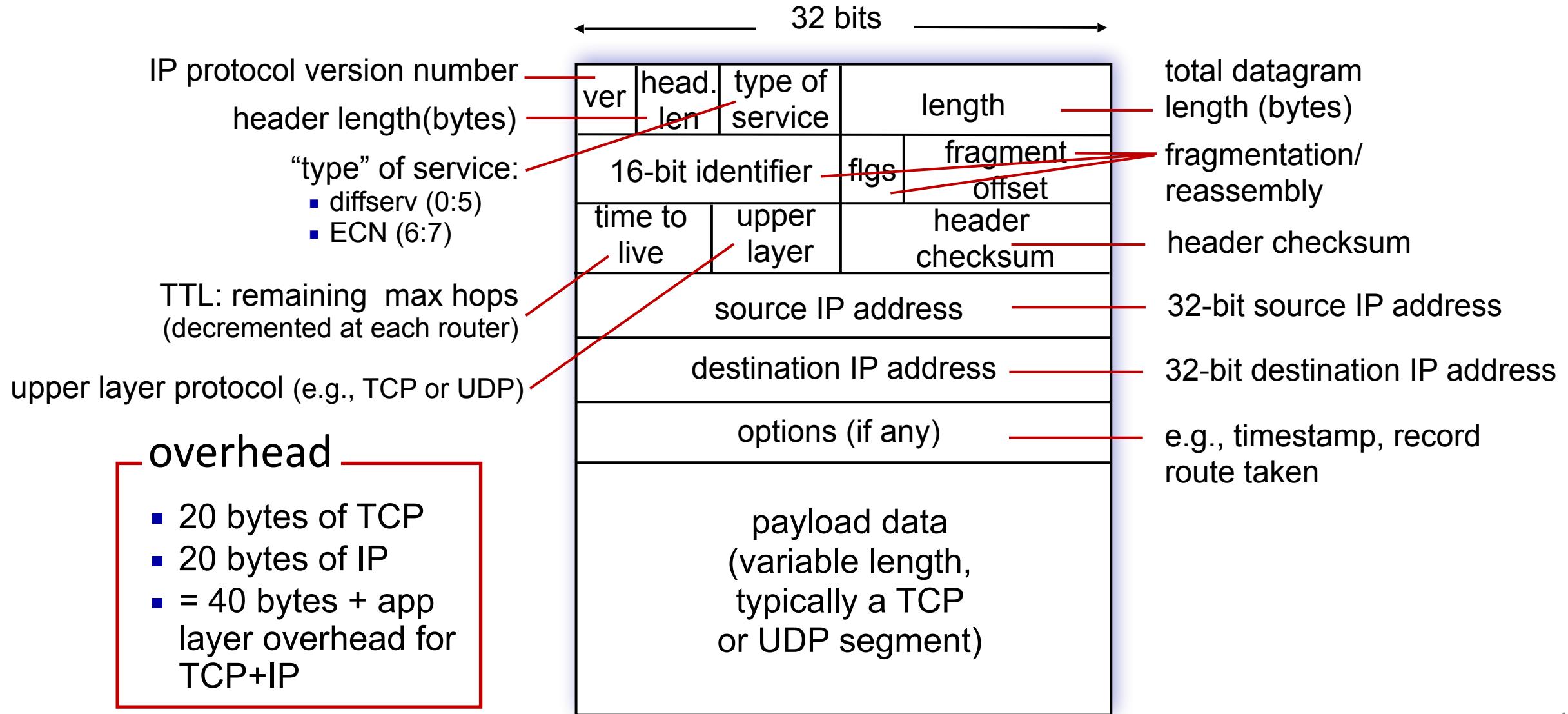
IP Datagram format



IP Datagram format



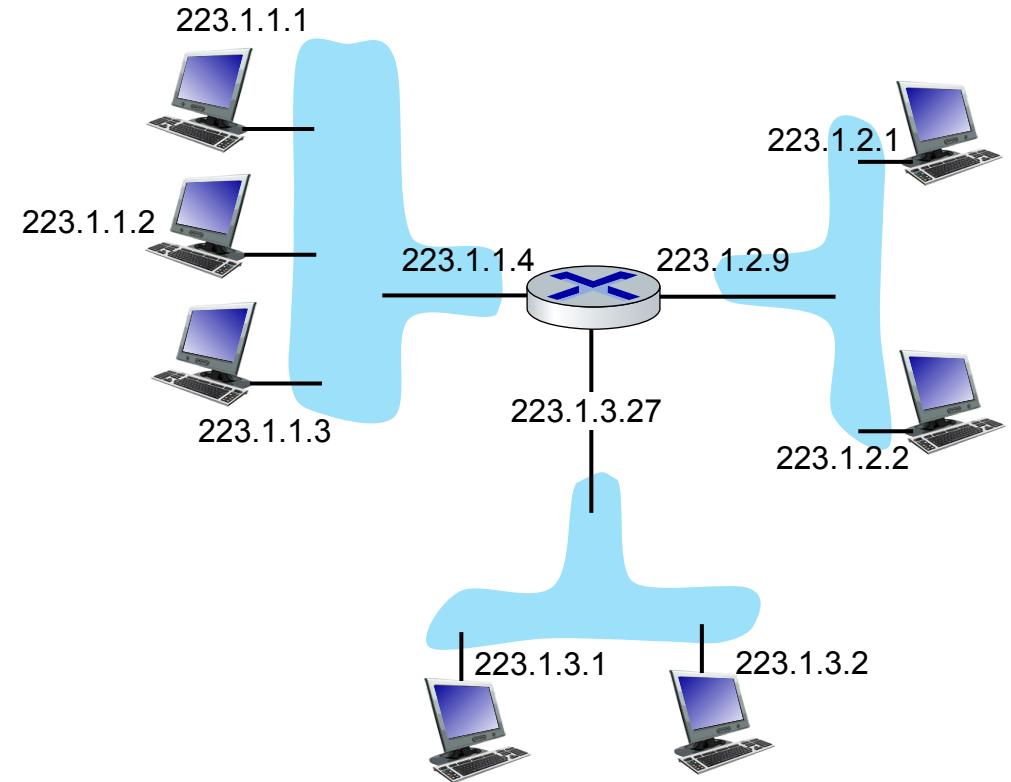
IP Datagram format



Chapter 4: roadmap

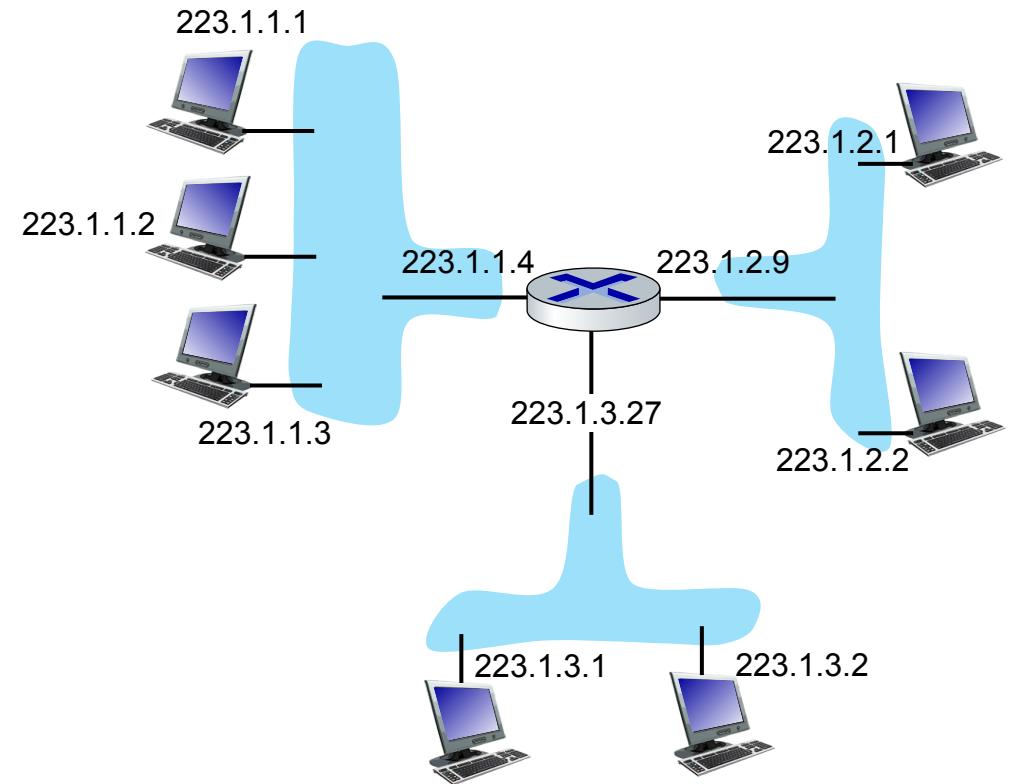
- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

IPv4 addressing: introduction



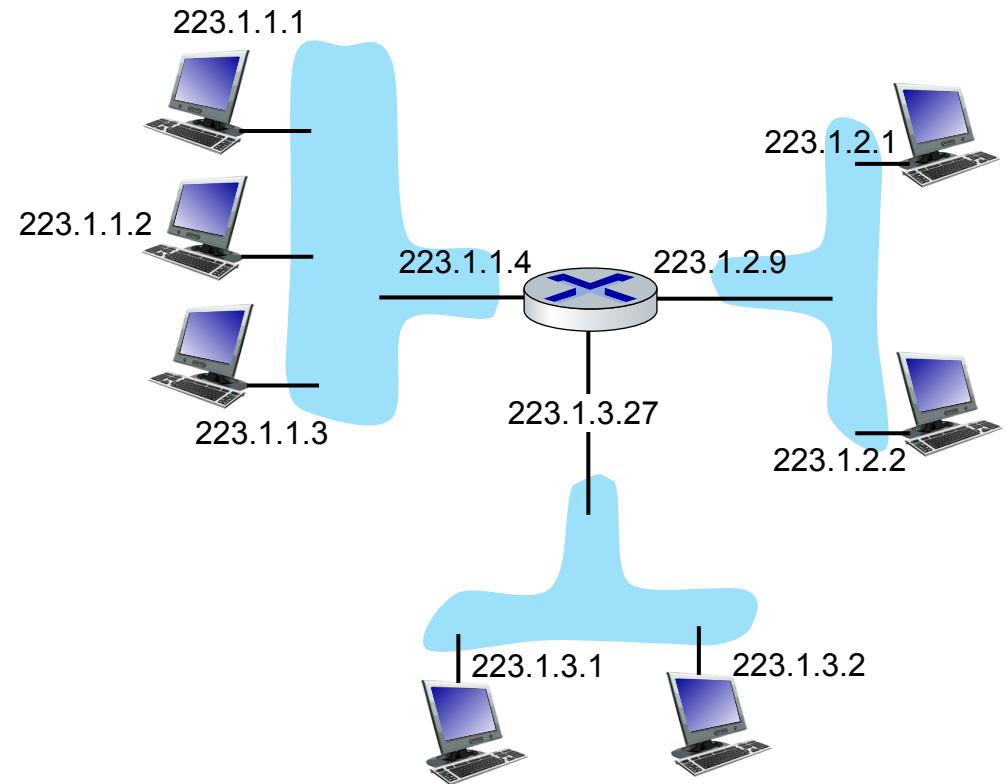
IPv4 addressing: introduction

- **IPv4 address:** 32-bit identifier associated with each host or router *interface*



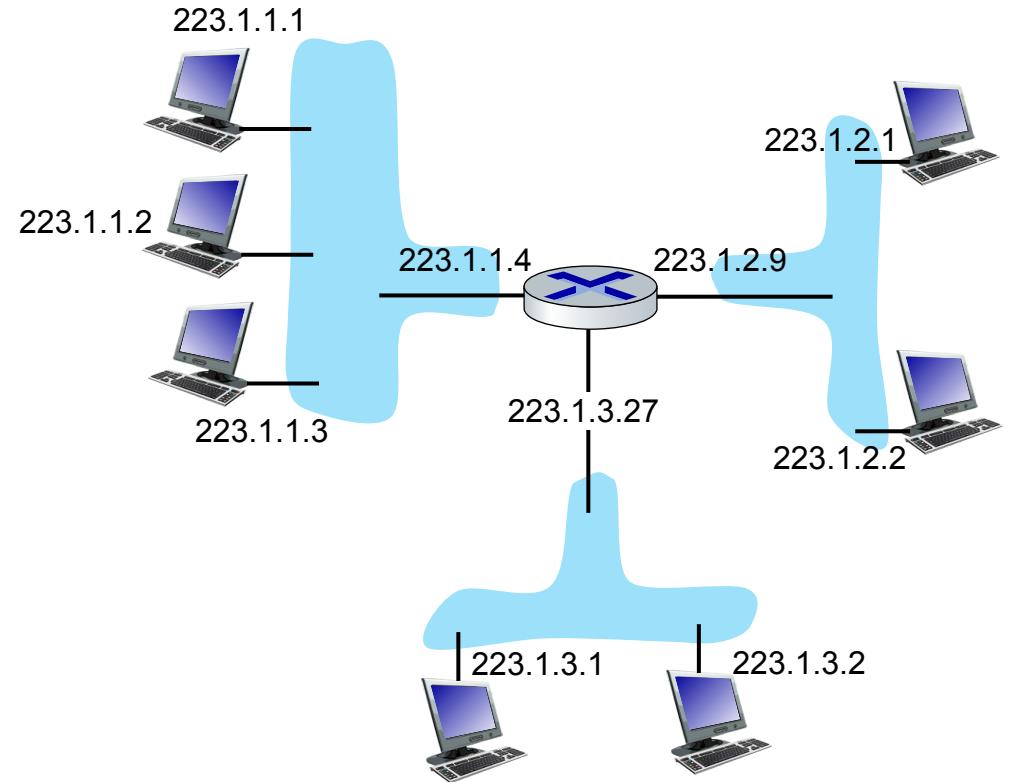
IPv4 addressing: introduction

- **IPv4 address:** 32-bit identifier associated with each host or router *interface*
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



IPv4 addressing: introduction

- **IPv4 address:** 32-bit identifier associated with each host or router *interface*
- **interface:** connection between host/router and physical link
 - router's typically have multiple interfaces
 - host typically has one or two interfaces (e.g., wired Ethernet, wireless 802.11)



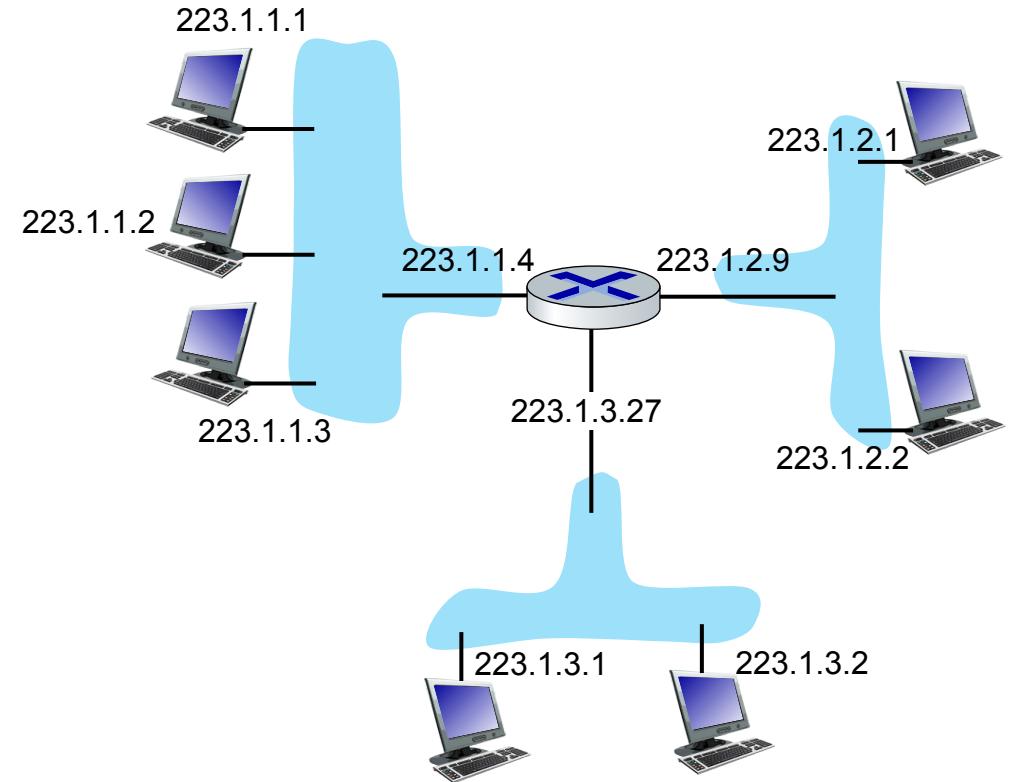
dotted-decimal IP address notation:

223.1.1.1 = 11011111 00000001 00000001 00000001

 | | | |
 223 1 1 1

IPv4 addressing: introduction

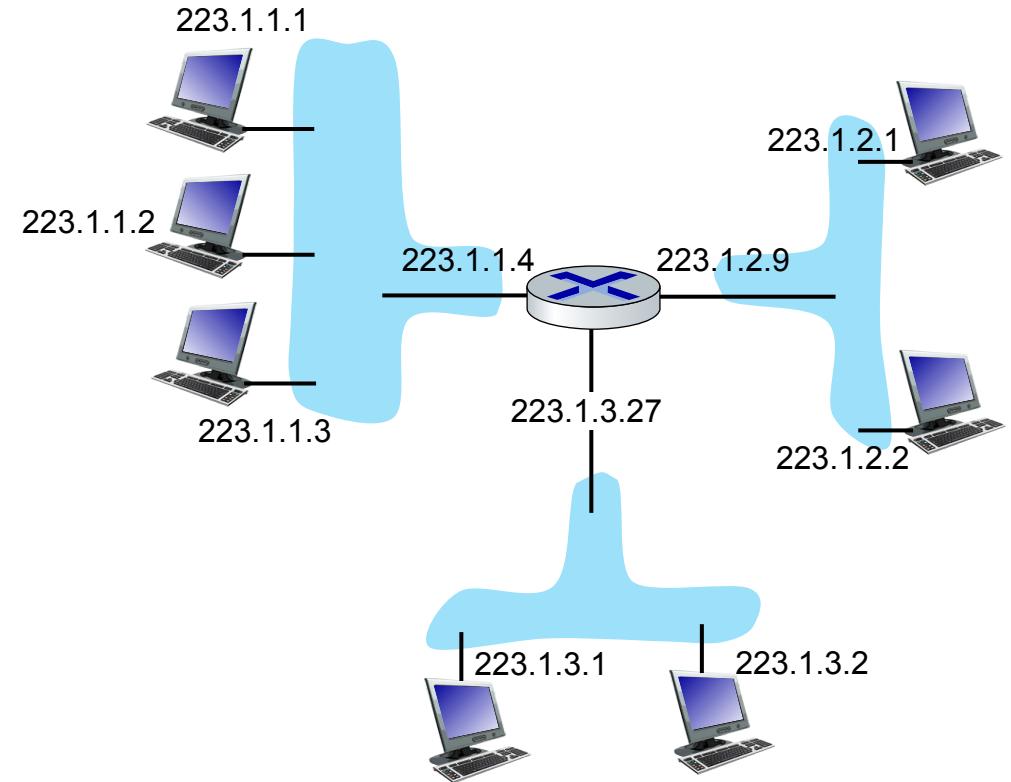
Q: how are interfaces
actually connected?



IPv4 addressing: introduction

Q: how are interfaces
actually connected?

A: we'll learn about
that in chapters 5, 6

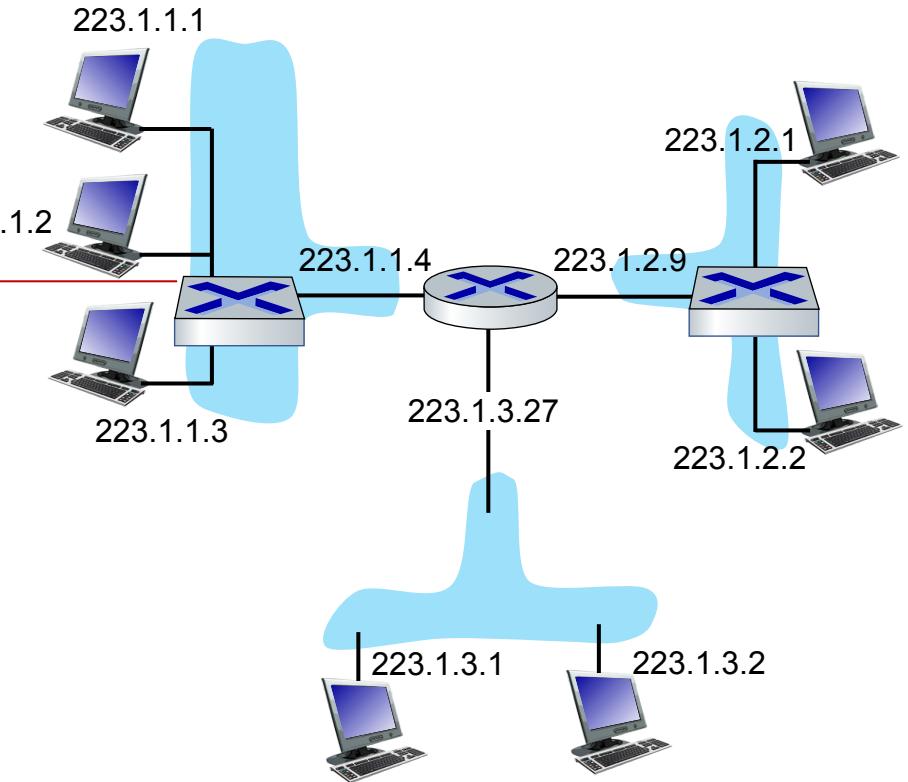


IPv4 addressing: introduction

Q: how are interfaces
actually connected?

A: we'll learn about
that in chapters 5, 6

A: wired
Ethernet interfaces
connected by
Ethernet switches

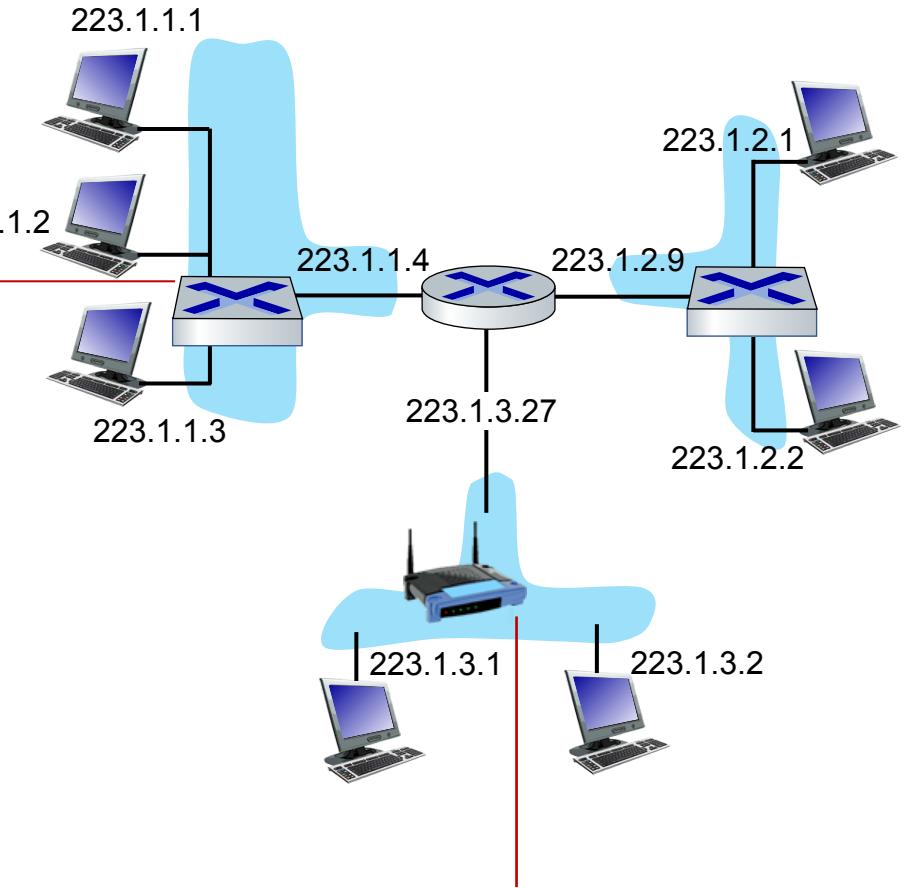


IPv4 addressing: introduction

Q: how are interfaces
actually connected?

A: we'll learn about
that in chapters 5, 6

A: wired
Ethernet interfaces
connected by
Ethernet switches



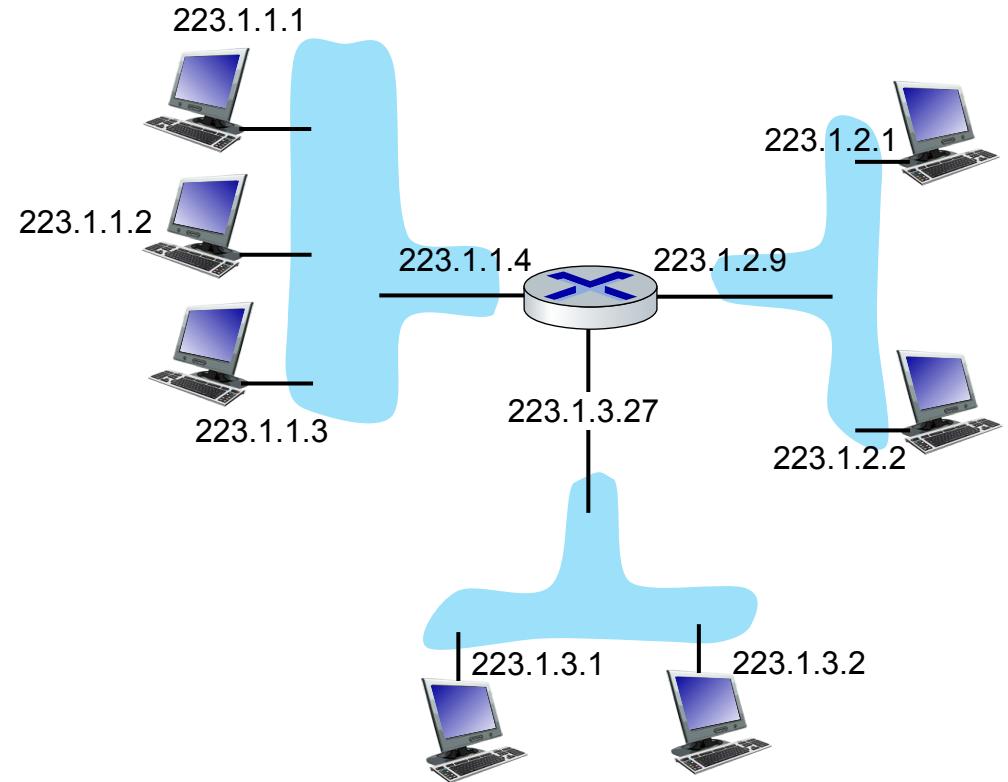
A: wireless WiFi interfaces
connected by WiFi base station

IPv4 addressing: introduction

Q: how are interfaces
actually connected?

A: we'll learn about
that in chapters 5, 6

For now: don't need to worry
about how one interface is
connected to another (with no
intervening router)



IPv4 addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: $a.b.c.d/x$, where x is # bits in subnet portion of address

IPv4 addressing: CIDR

CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



IPv4 addressing: CIDR

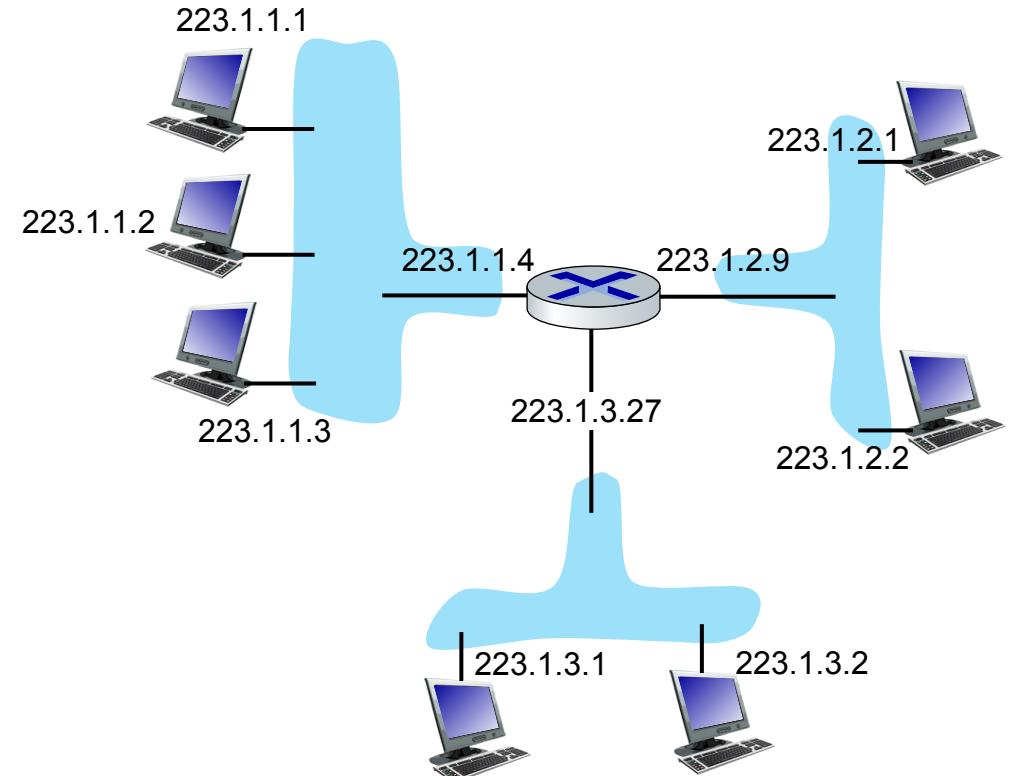
CIDR: Classless InterDomain Routing (pronounced “cider”)

- subnet portion of address of arbitrary length
- address format: **a.b.c.d/x**, where x is # bits in subnet portion of address



- when **host part is all-ones**, it represents the **broadcast address** of that subnet

Subnets

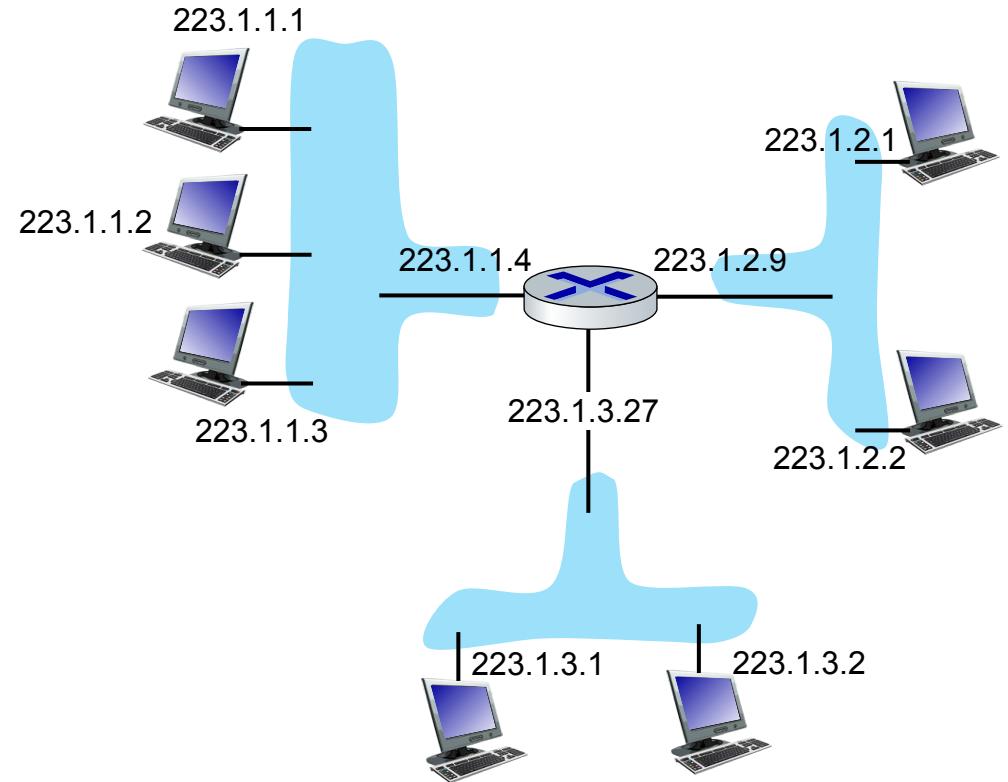


network consisting of 3 subnets

Subnets

■ *What's a subnet ?*

- device interfaces that can physically reach each other **without passing through an intervening router**



network consisting of 3 subnets

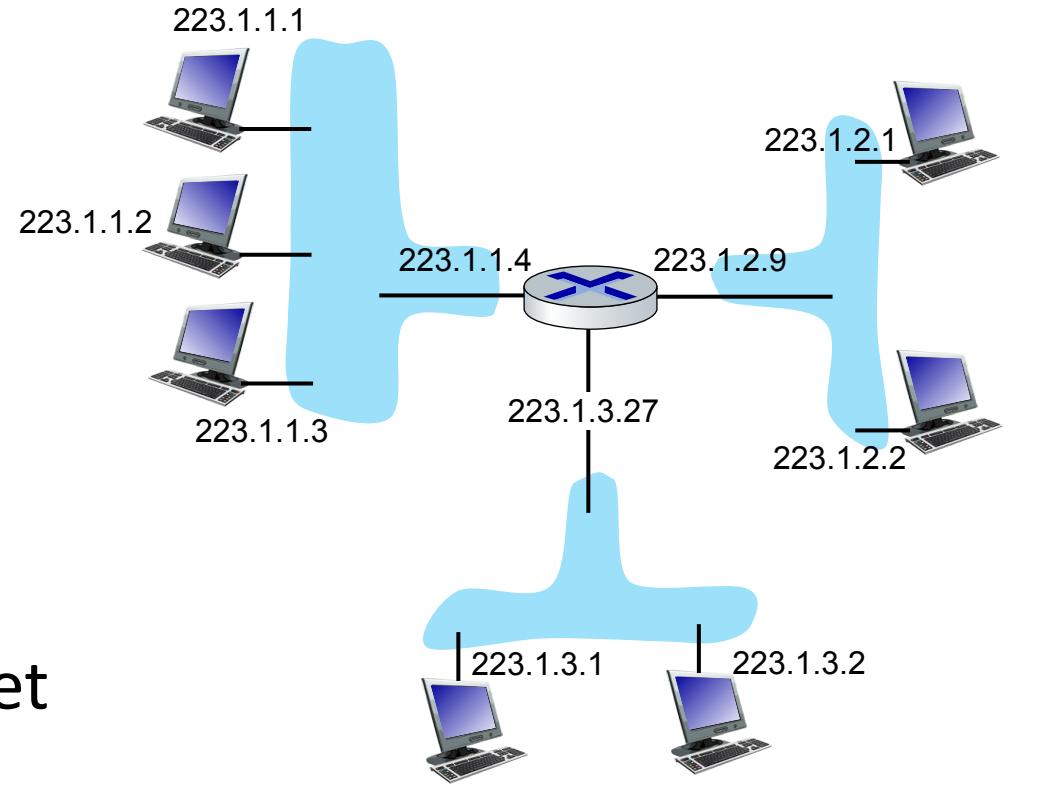
Subnets

■ *What's a subnet ?*

- device interfaces that can physically reach each other **without passing through an intervening router**

■ IP addresses have structure:

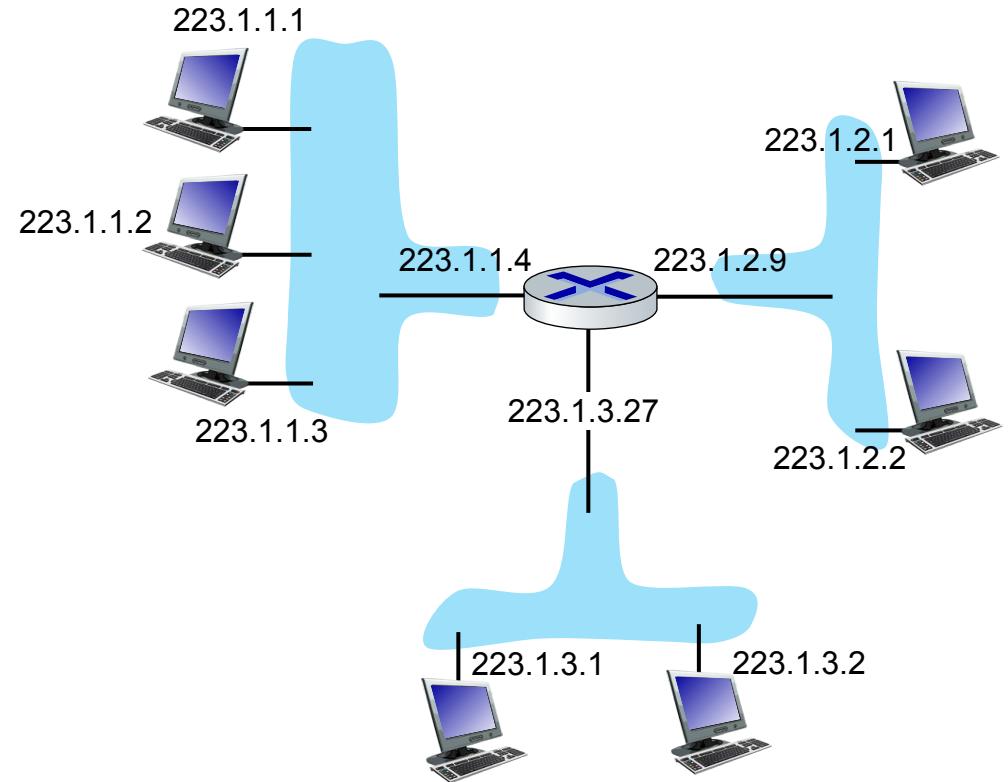
- **subnet part:** devices in same subnet have common high order bits
- **host part:** remaining low order bits



network consisting of 3 subnets

Subnets

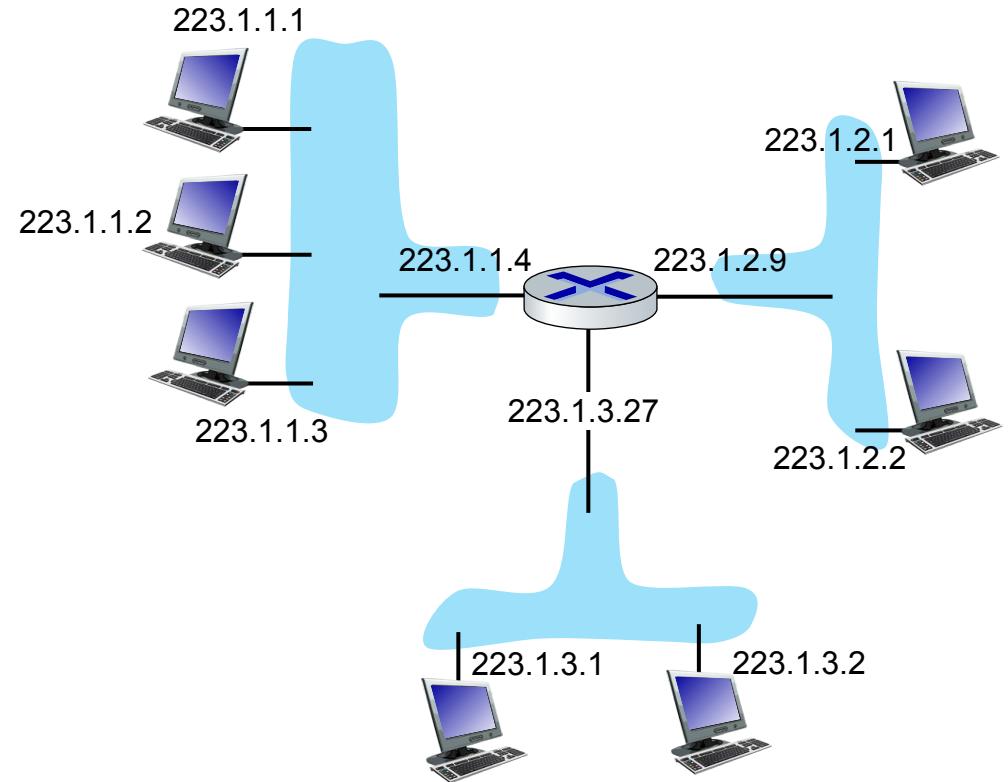
Recipe for defining subnets:



Subnets

Recipe for defining subnets:

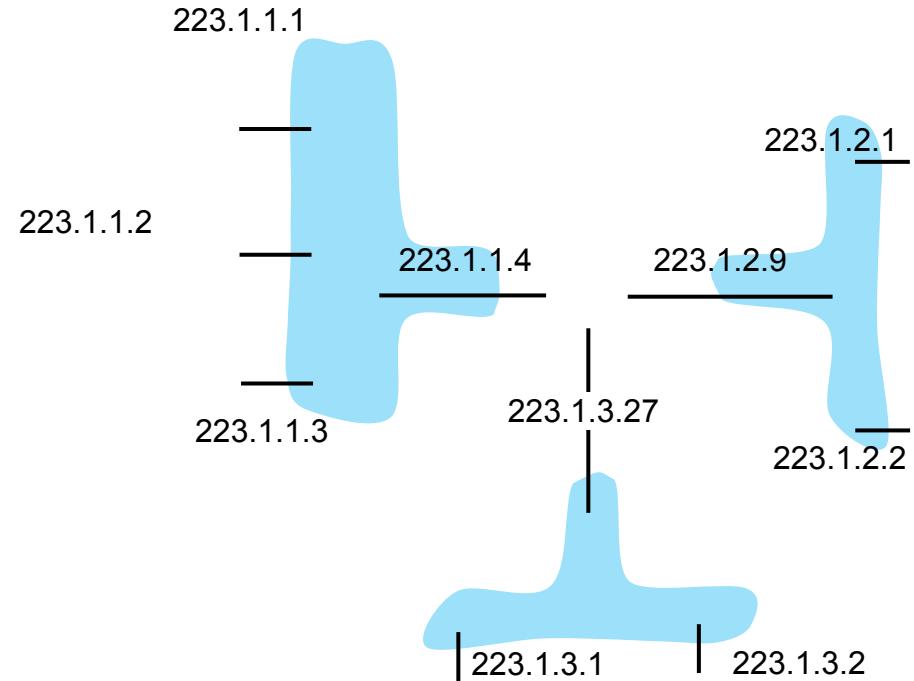
- detach each interface from its host or router, creating “islands” of isolated networks



Subnets

Recipe for defining subnets:

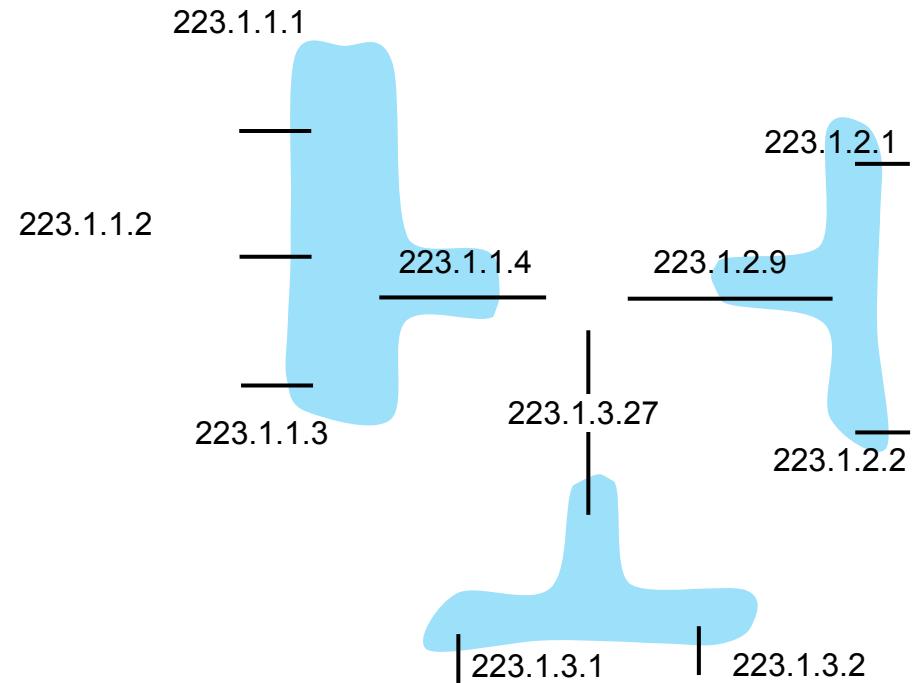
- detach each interface from its host or router, creating “islands” of isolated networks



Subnets

Recipe for defining subnets:

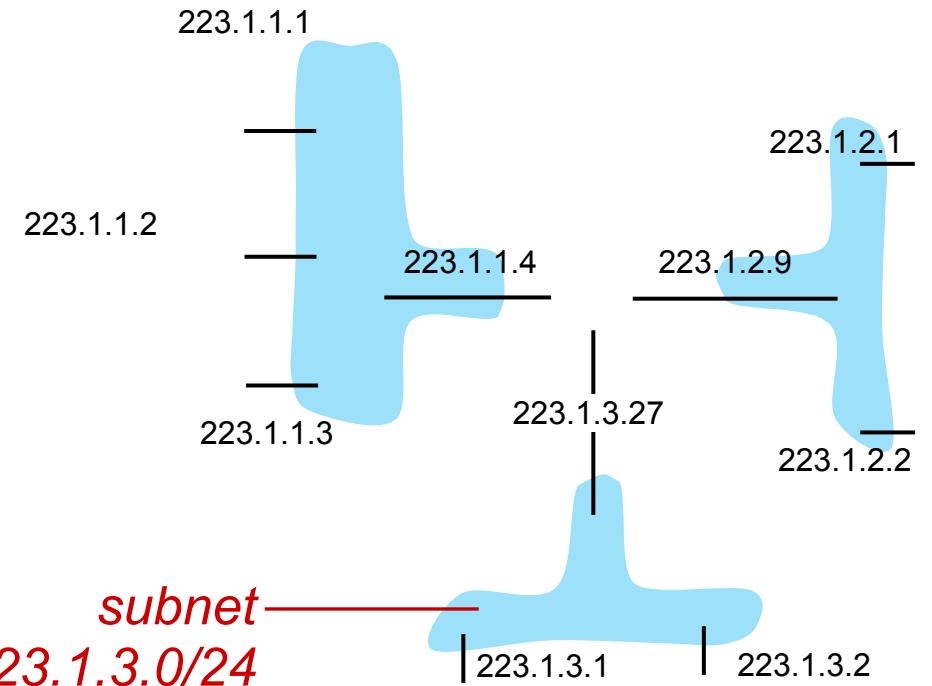
- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*



Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*

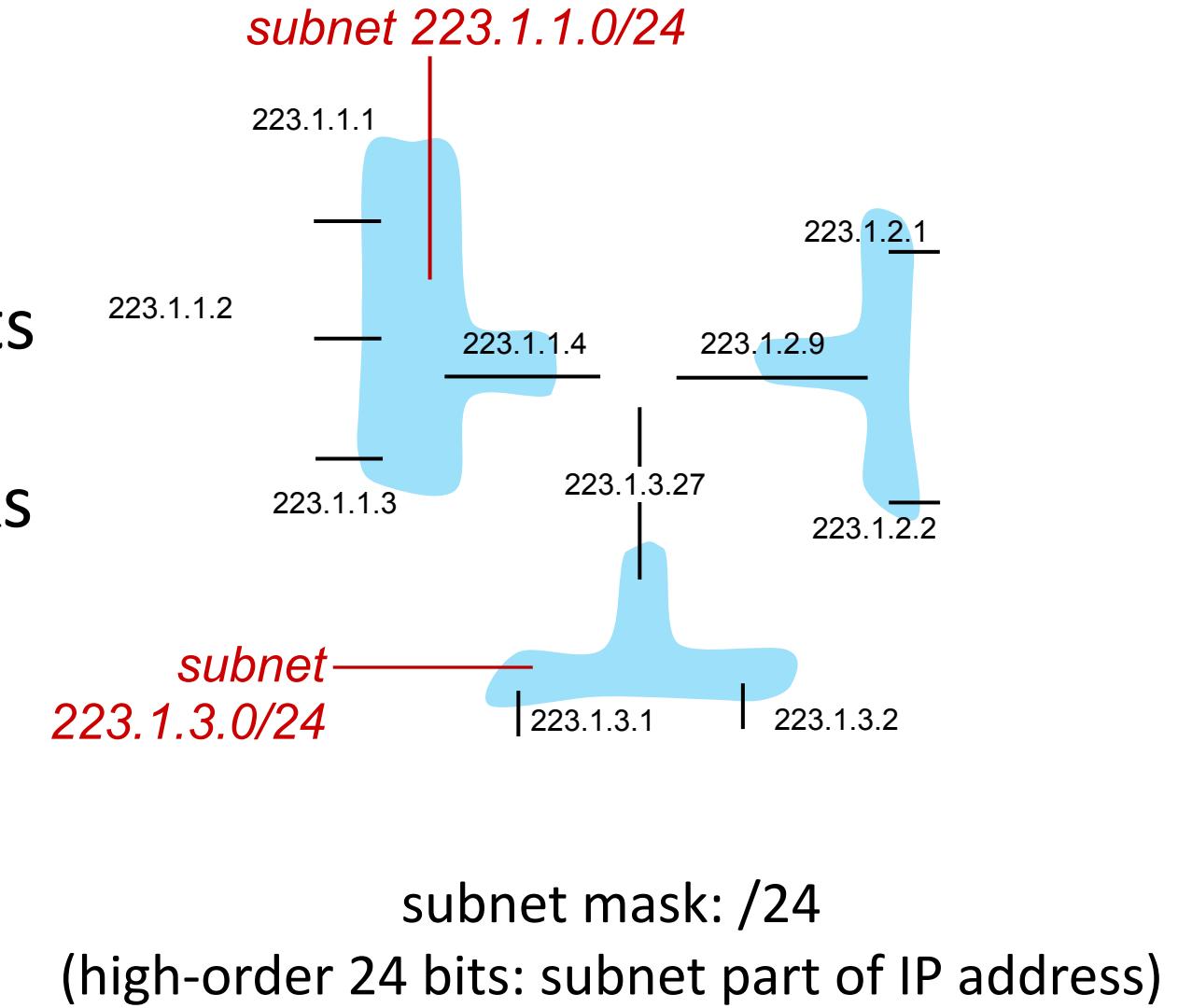


subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

Recipe for defining subnets:

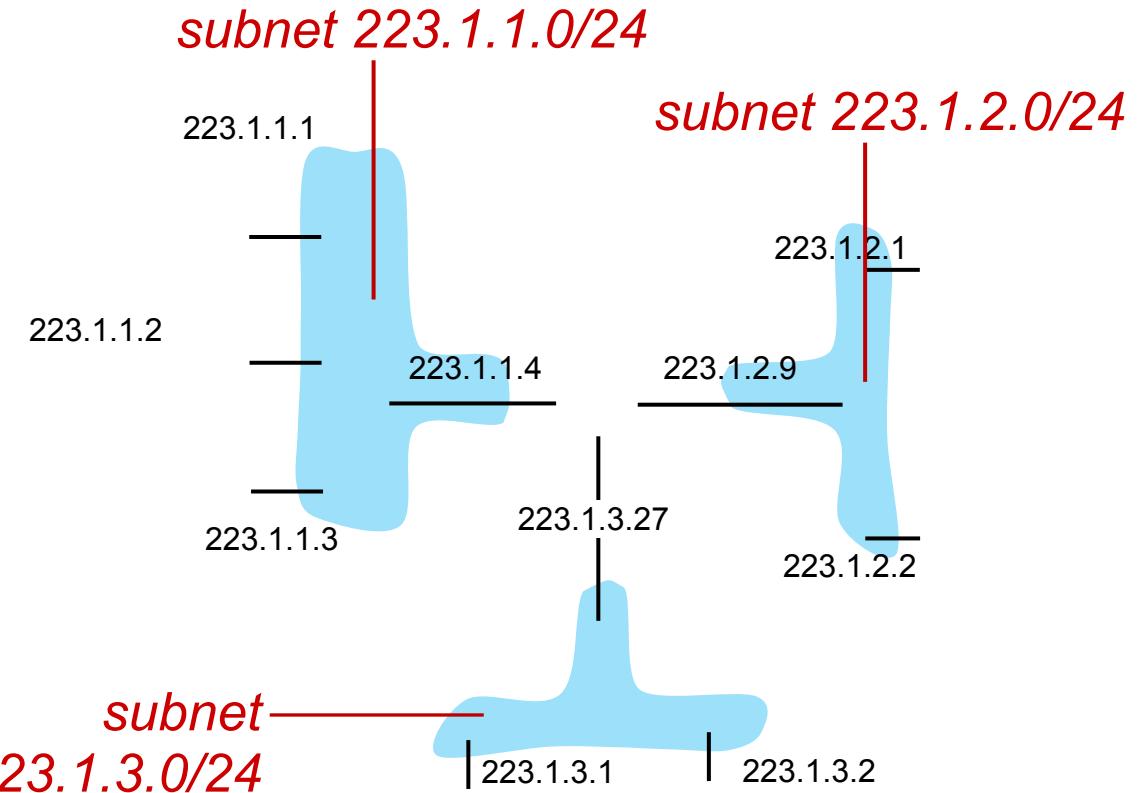
- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*



Subnets

Recipe for defining subnets:

- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*

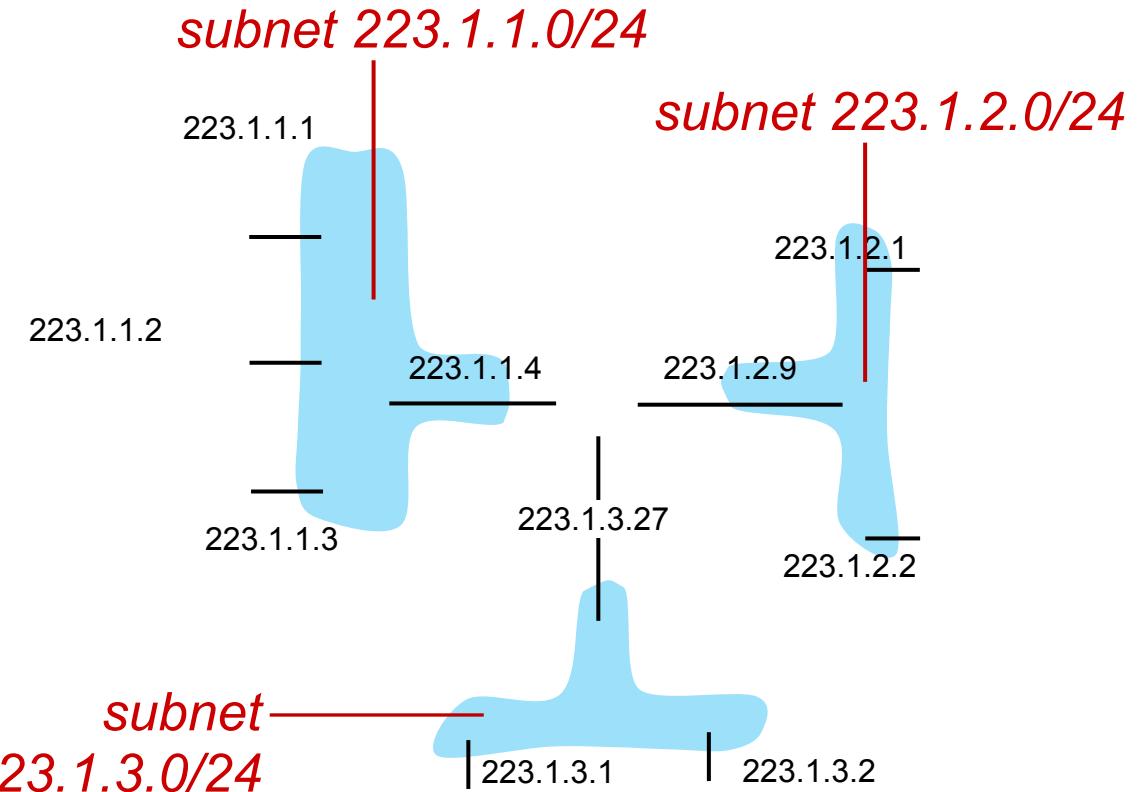


subnet mask: /24
(high-order 24 bits: subnet part of IP address)

Subnets

Recipe for defining subnets:

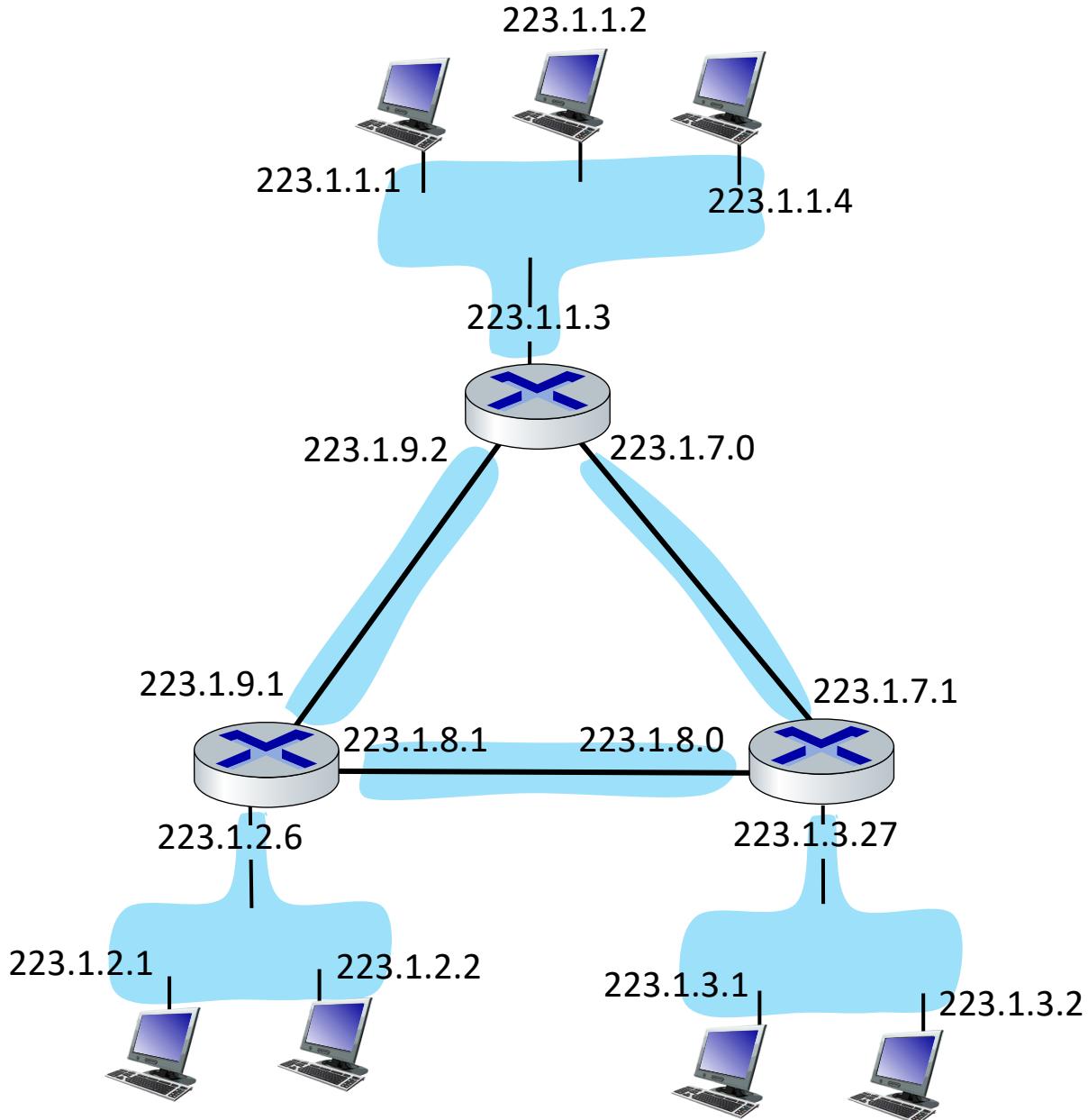
- detach each interface from its host or router, creating “islands” of isolated networks
- each isolated network is called a *subnet*



subnet mask: /24
(high-order 24 bits: subnet part of IP address)

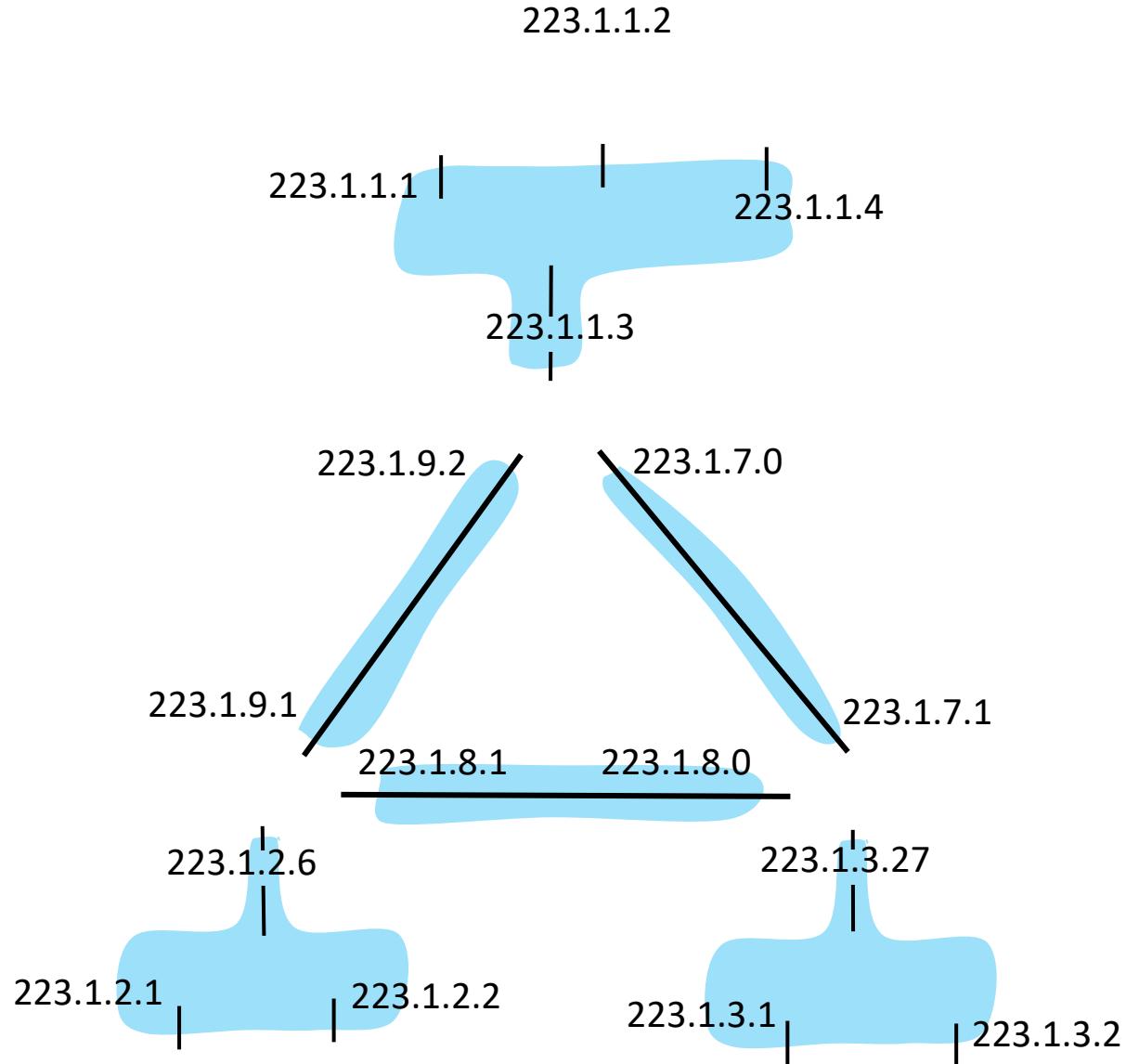
Subnets

- where are the subnets?
- what are the /24 subnet addresses?



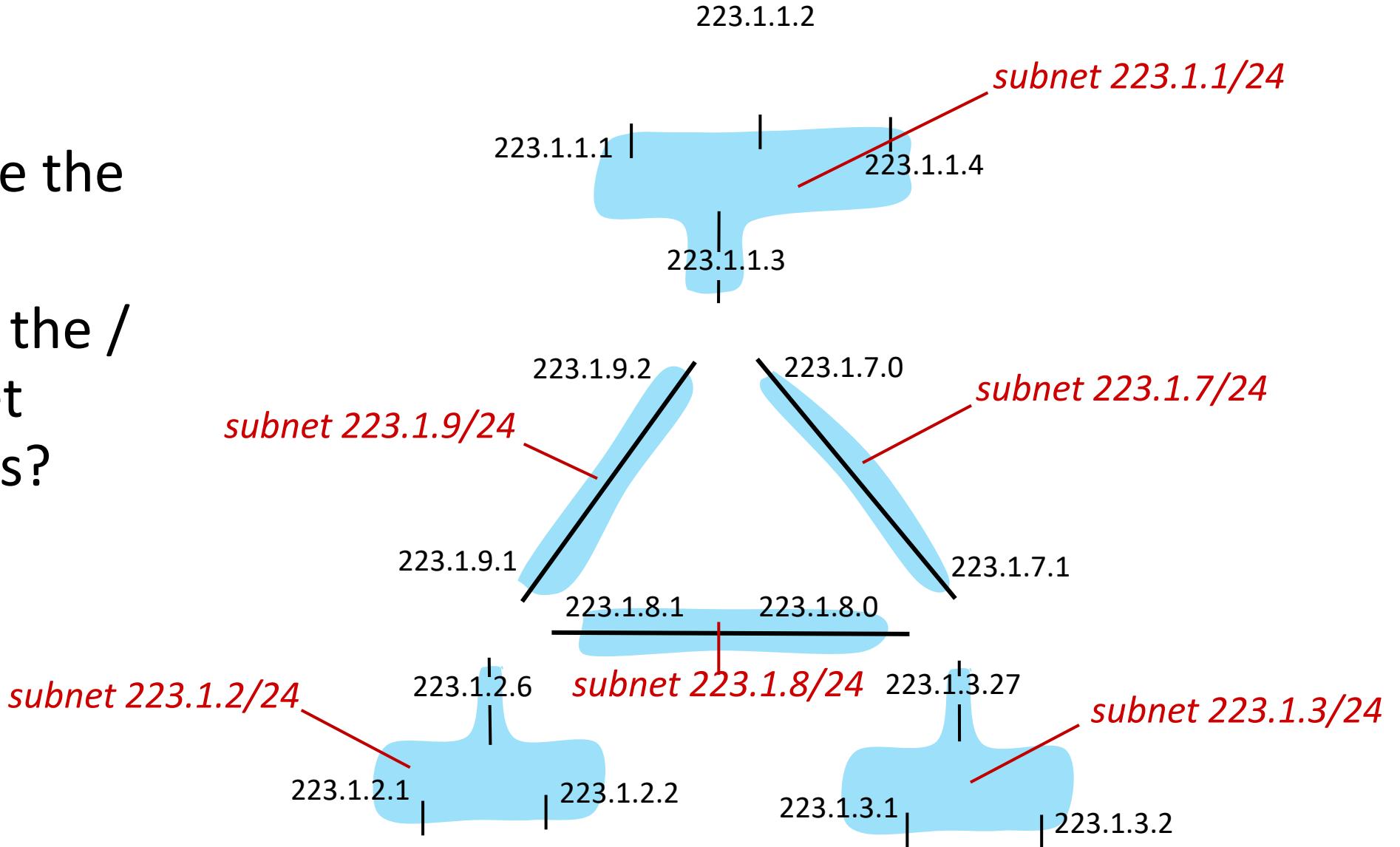
Subnets

- where are the subnets?
- what are the /24 subnet addresses?



Subnets

- where are the subnets?
- what are the /24 subnet addresses?



IPv4 addresses: special addresses

- 0.0.0.0/8 – Used as source address for this host on this network
- 100.64.0.0/10 – Used for *Carrier Grade NAT*
- 127.0.0.0/8 – Used as loopback, usually it is 127.0.0.1
- 169.254.0.0/8 – Link local, dynamically generated
- 192.0.0.0/24 – Used for protocol assignments
- 192.0.2.0/24, 198.51.100.0/24 and 203.0.113.0/24 – Used in documentation and example code
 - Used in conjunction with “example.com” or “example.net”
- 192.88.99.0/24 – Used in 6to4 relay anycast
 - Forwarded on the Internet
- 198.18.0.0/15 – Used for benchmark tests
- 224.0.0.0/4 – Used in multicast
 - Only as destination address
 - Forwarded on the Internet
- 240.0.0.0/4 – Reserved for future use
- 255.255.255.255/32 – Limited broadcast

IPv4 addresses: private addresses

Used in private networks:

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

IP addresses: how to get one?

That's actually **two** questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

IP addresses: how to get one?

That's actually **two** questions:

1. Q: How does a *host* get IP address within its network (host part of address)?
2. Q: How does a *network* get IP address for itself (network part of address)

How does *host* get IP address?

- hard-coded by sysadmin in config file (e.g., `/etc/rc.config` in UNIX)
- **DHCP: Dynamic Host Configuration Protocol:** dynamically get address from server
 - “plug-and-play”

DHCP: Dynamic Host Configuration Protocol

goal: host *dynamically* obtains IP address from network server when it “joins” network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

DHCP: Dynamic Host Configuration Protocol

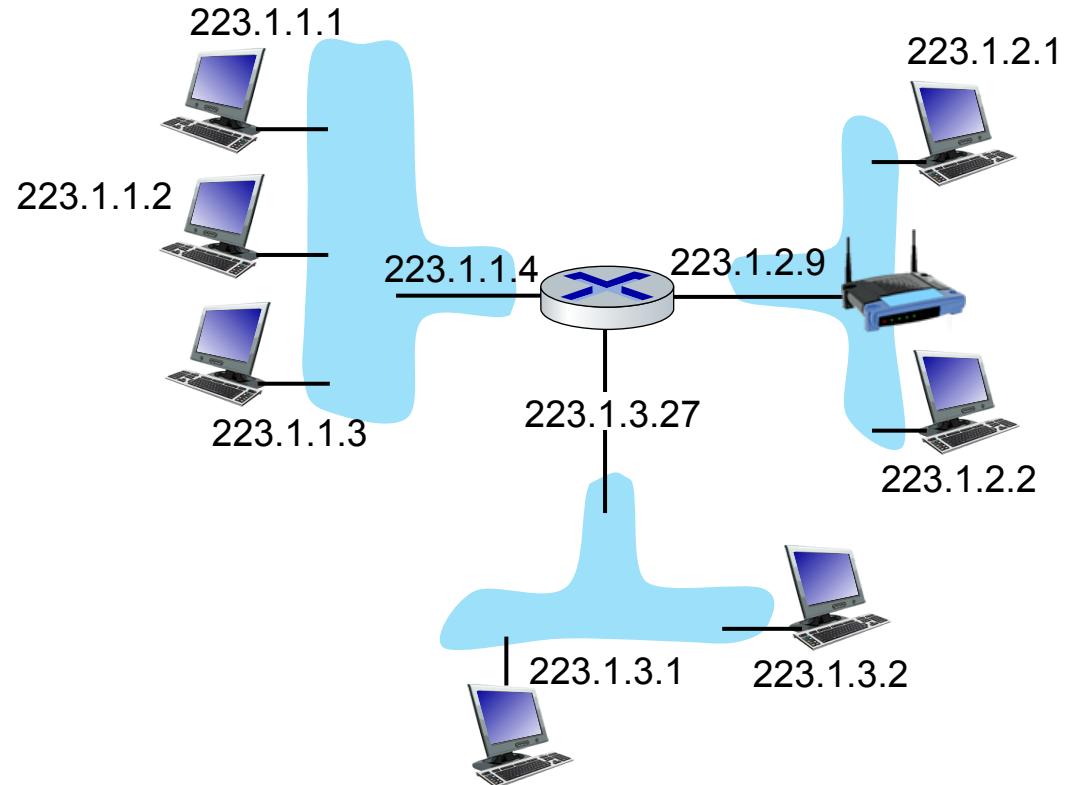
goal: host *dynamically* obtains IP address from network server when it “joins” network

- can renew its lease on address in use
- allows reuse of addresses (only hold address while connected/on)
- support for mobile users who join/leave network

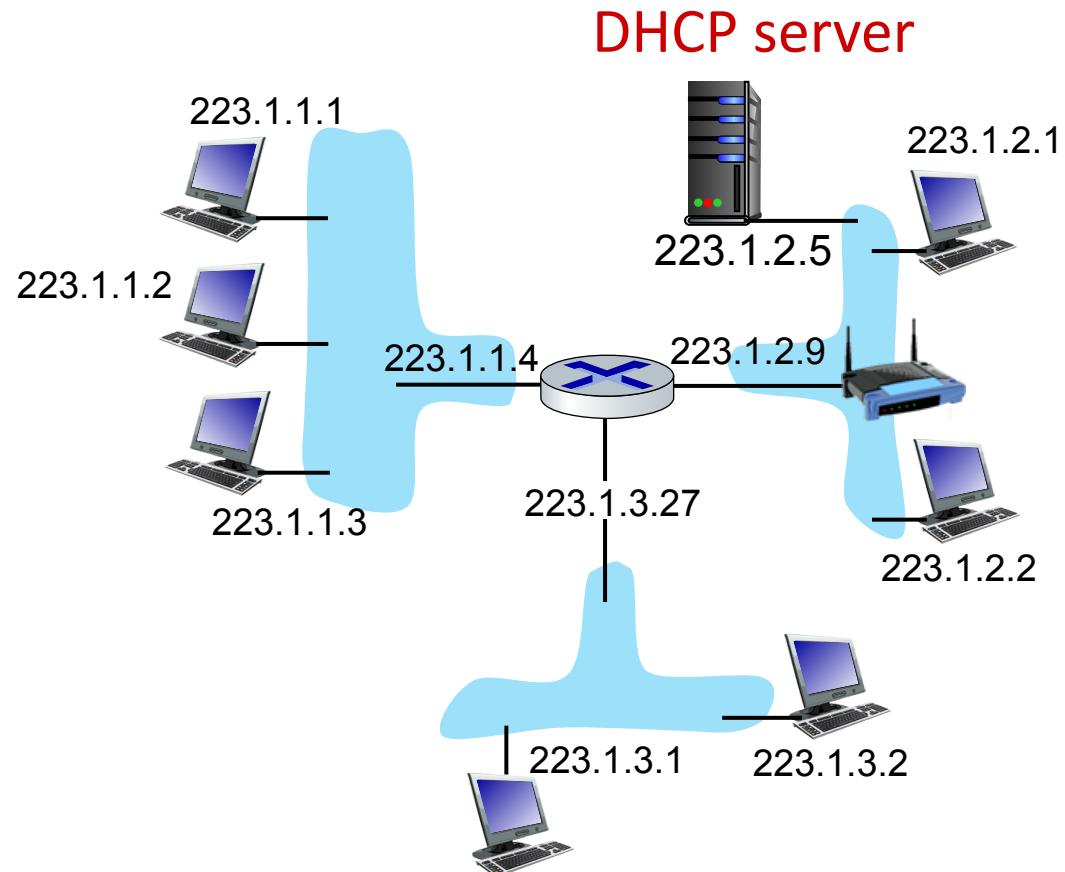
DHCP overview:

- host broadcasts **DHCP discover** msg [optional]
- DHCP server responds with **DHCP offer** msg [optional]
- host requests IP address: **DHCP request** msg
- DHCP server sends address: **DHCP ack** msg

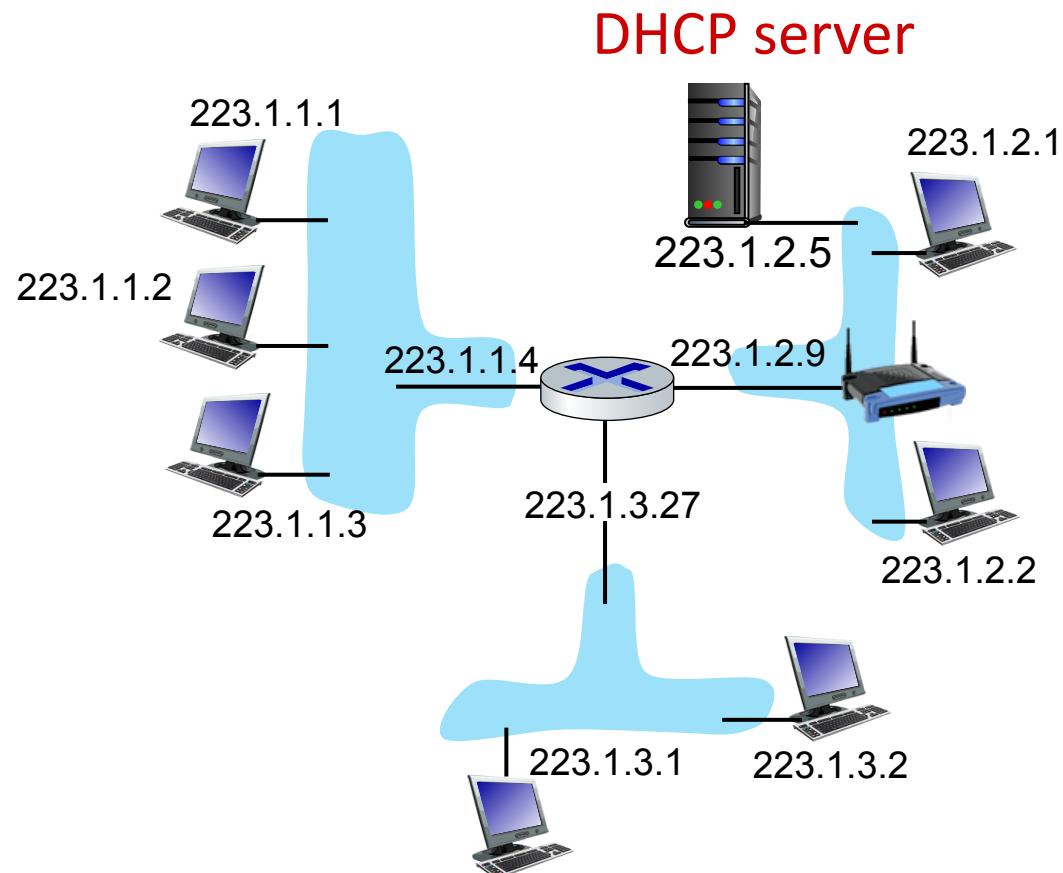
DHCP client-server scenario



DHCP client-server scenario

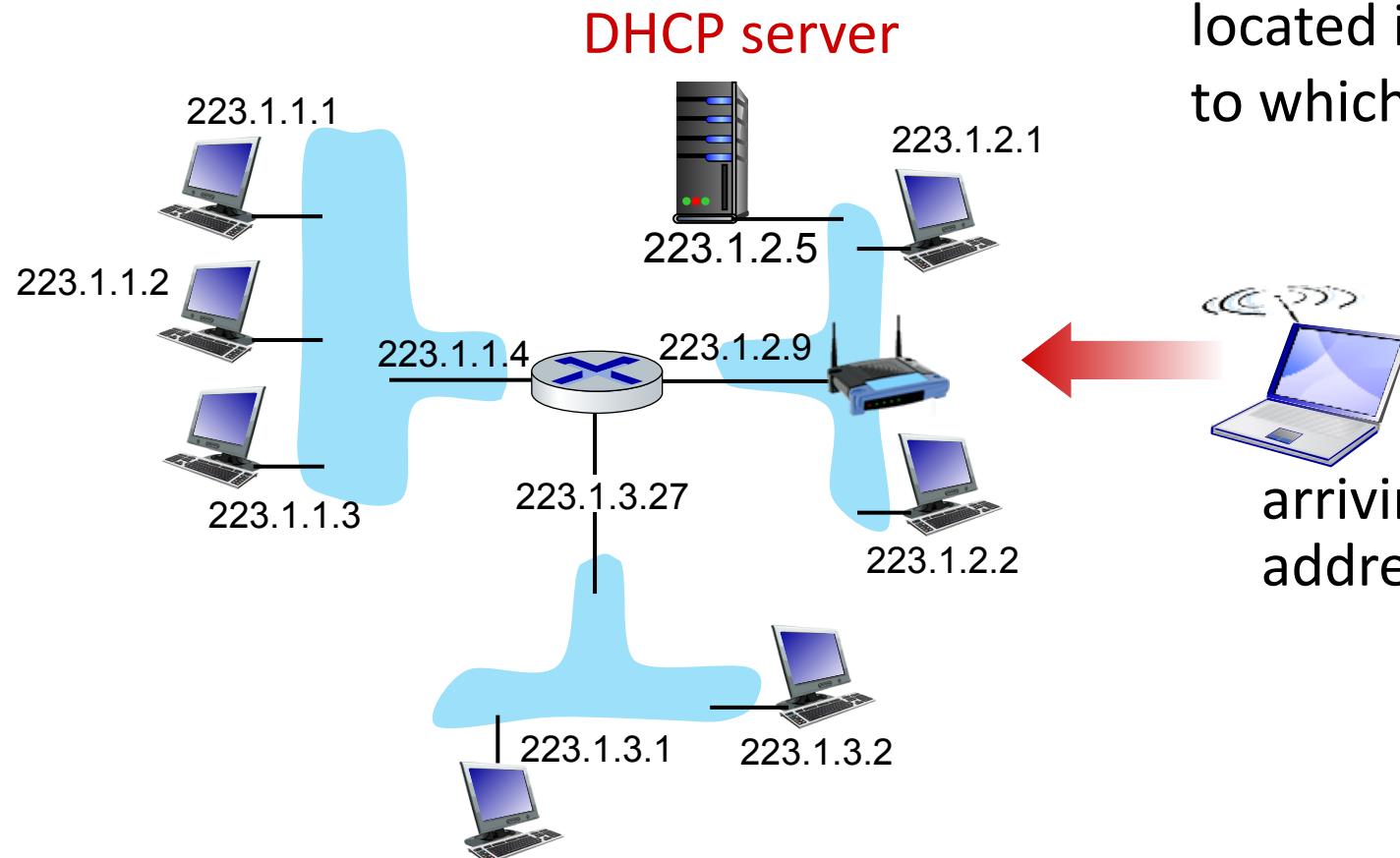


DHCP client-server scenario



Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

DHCP client-server scenario



Typically, DHCP server will be co-located in router, serving all subnets to which router is attached

arriving **DHCP client** needs address in this network

DHCP client-server scenario

DHCP server: 223.1.2.5



Arriving client



DHCP client-server scenario

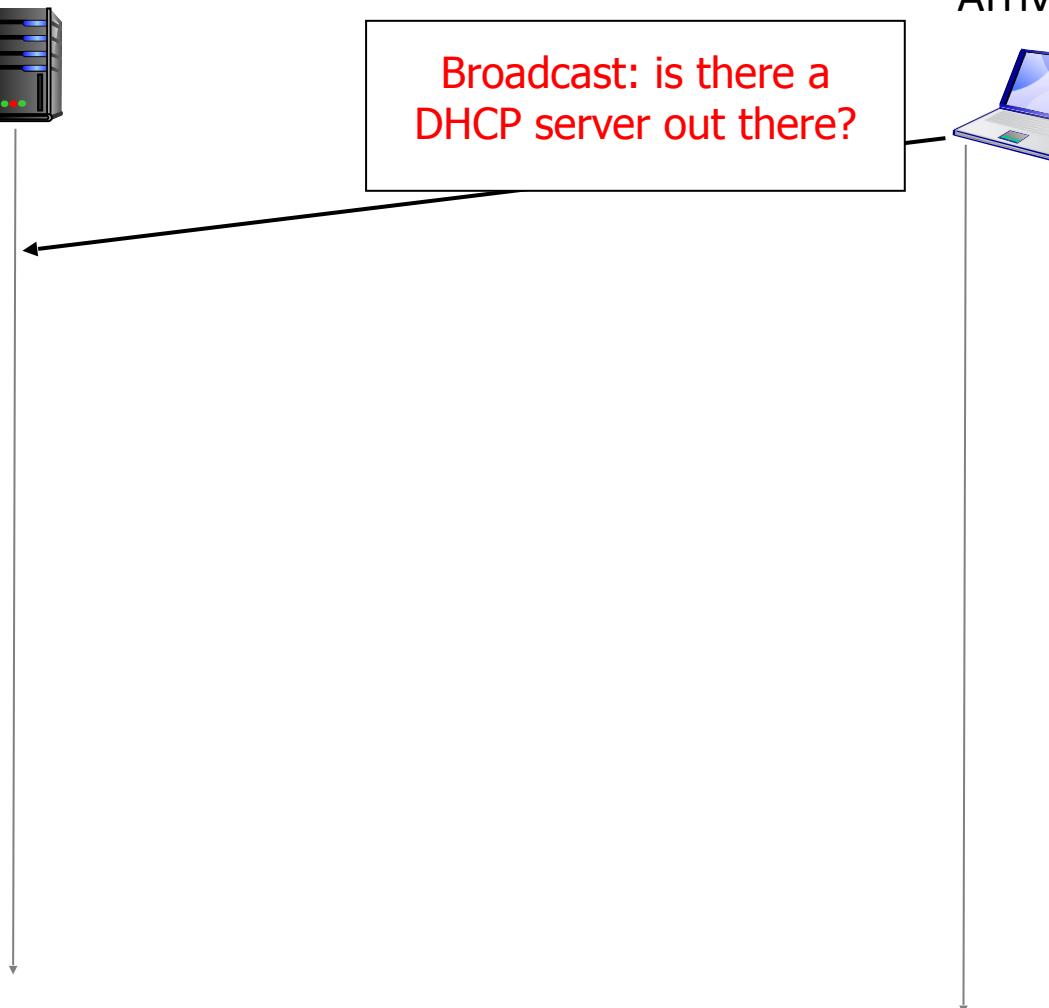
DHCP server: 223.1.2.5



DHCP discover

Broadcast: is there a
DHCP server out there?

Arriving client



DHCP client-server scenario

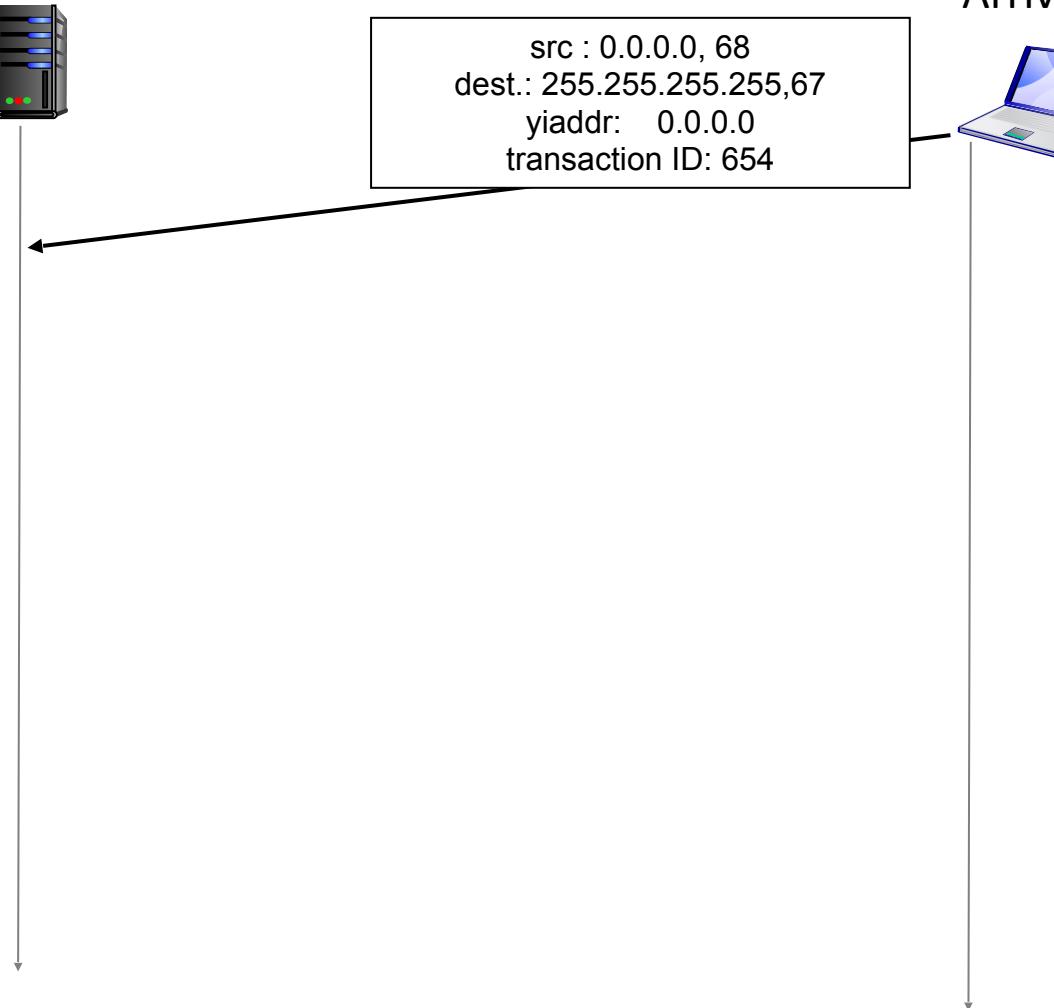
DHCP server: 223.1.2.5



DHCP discover

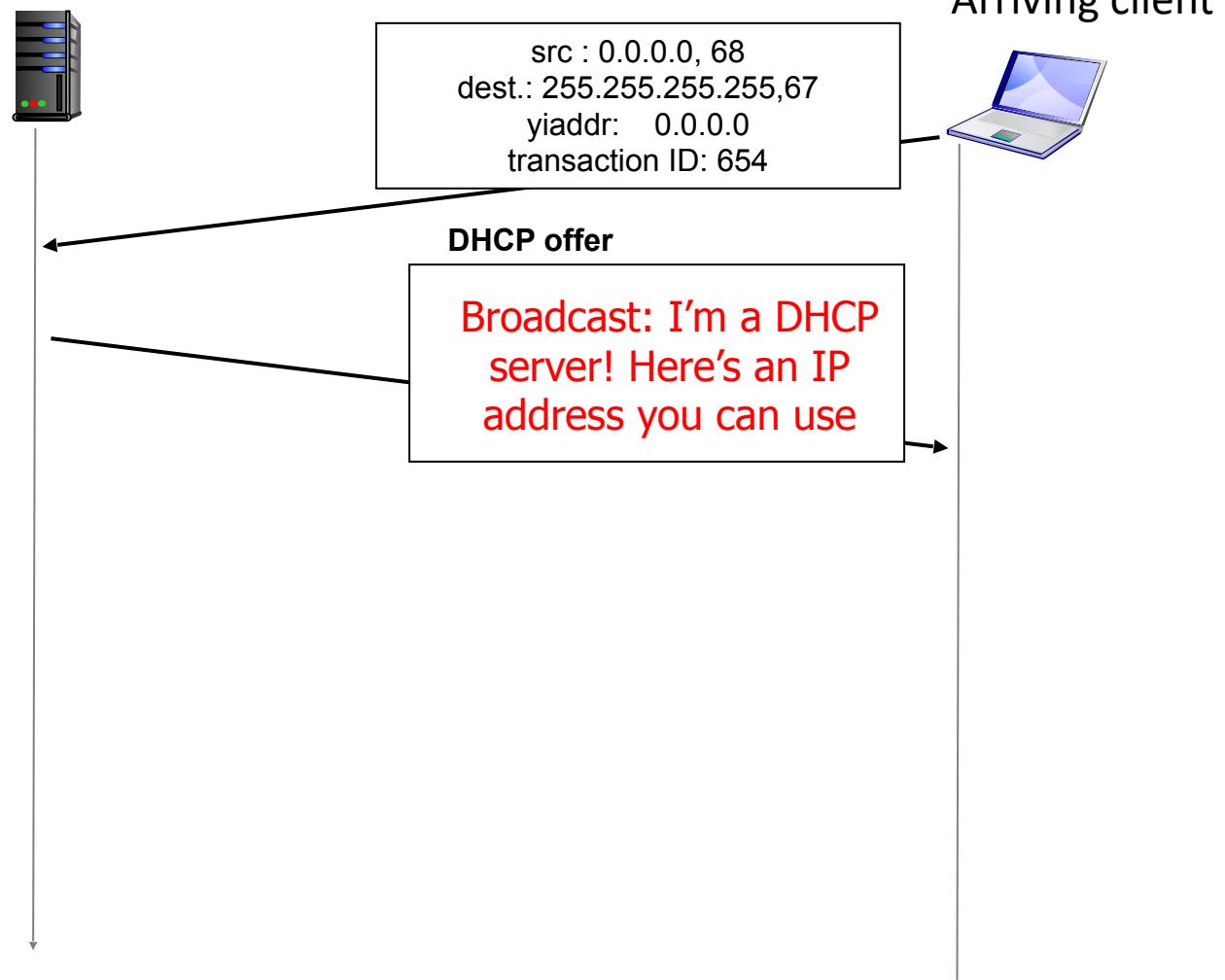
```
src : 0.0.0.0, 68  
dest.: 255.255.255.255,67  
yiaddr: 0.0.0.0  
transaction ID: 654
```

Arriving client



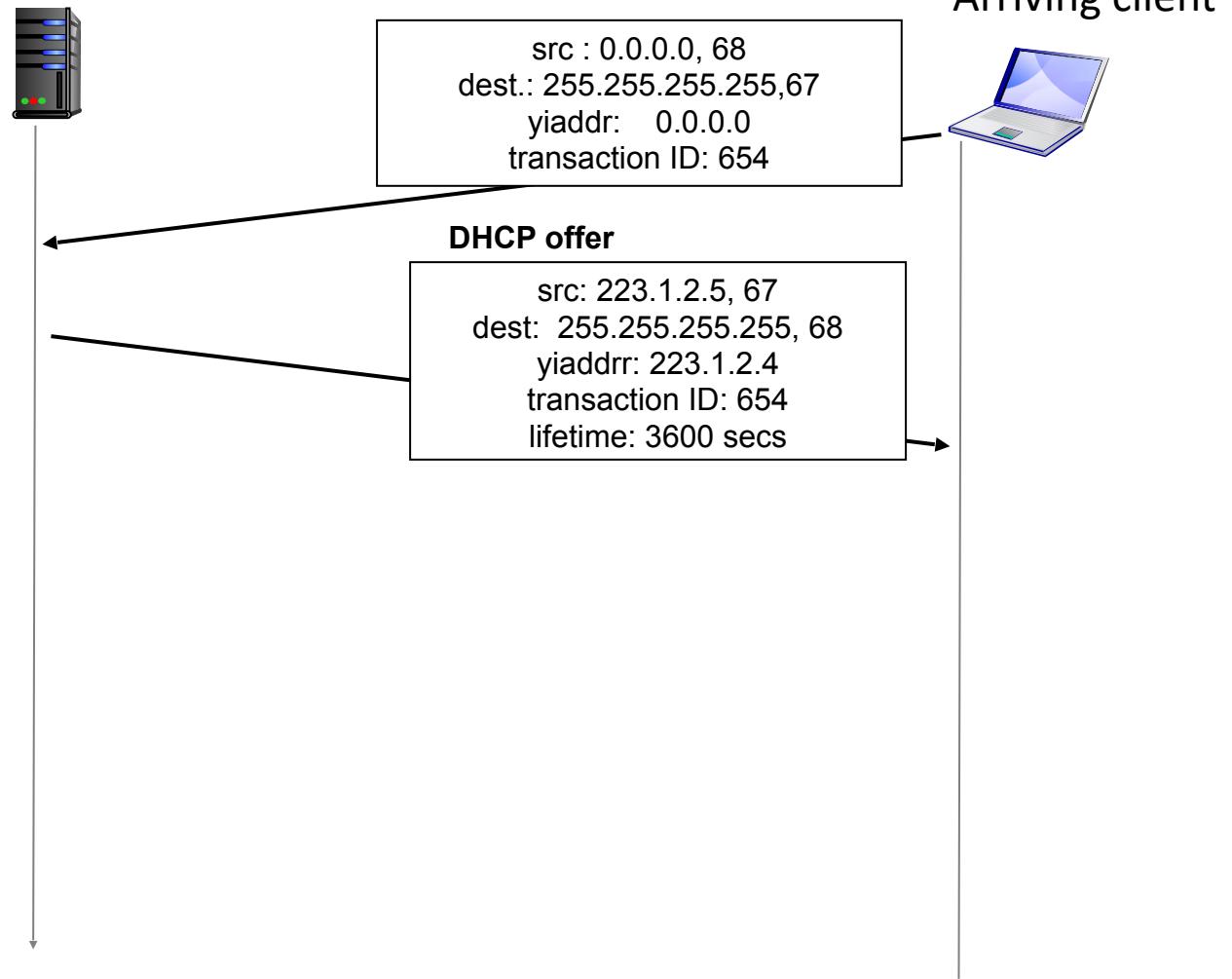
DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP client-server scenario

DHCP server: 223.1.2.5



DHCP discover

```
src : 0.0.0.0, 68  
dest.: 255.255.255.255,67  
yiaddr: 0.0.0.0  
transaction ID: 654
```

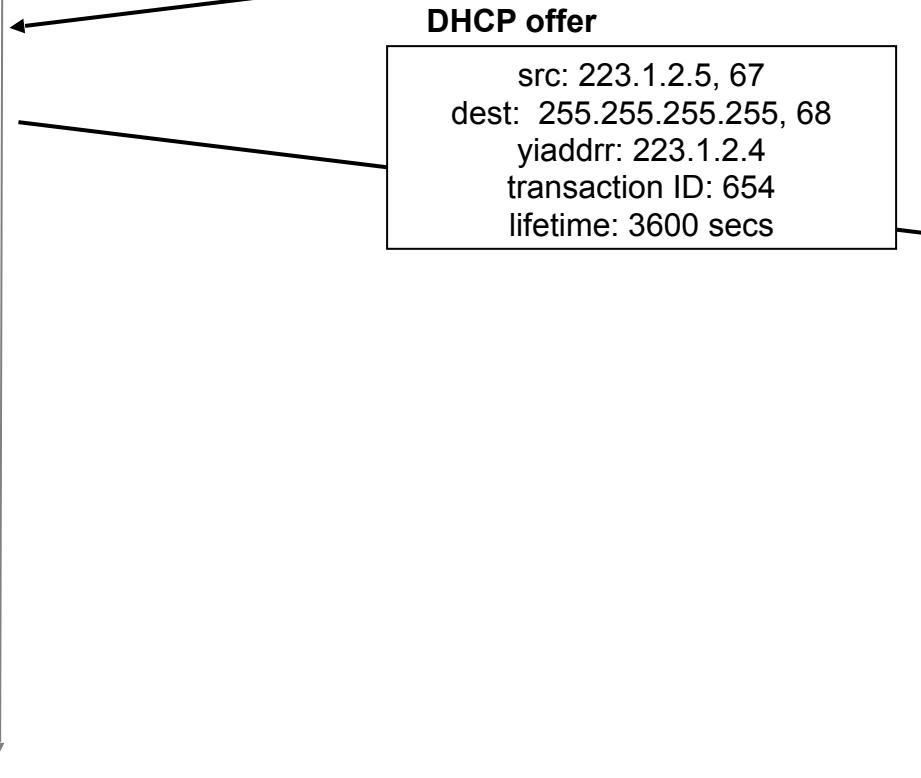
Arriving client



DHCP offer

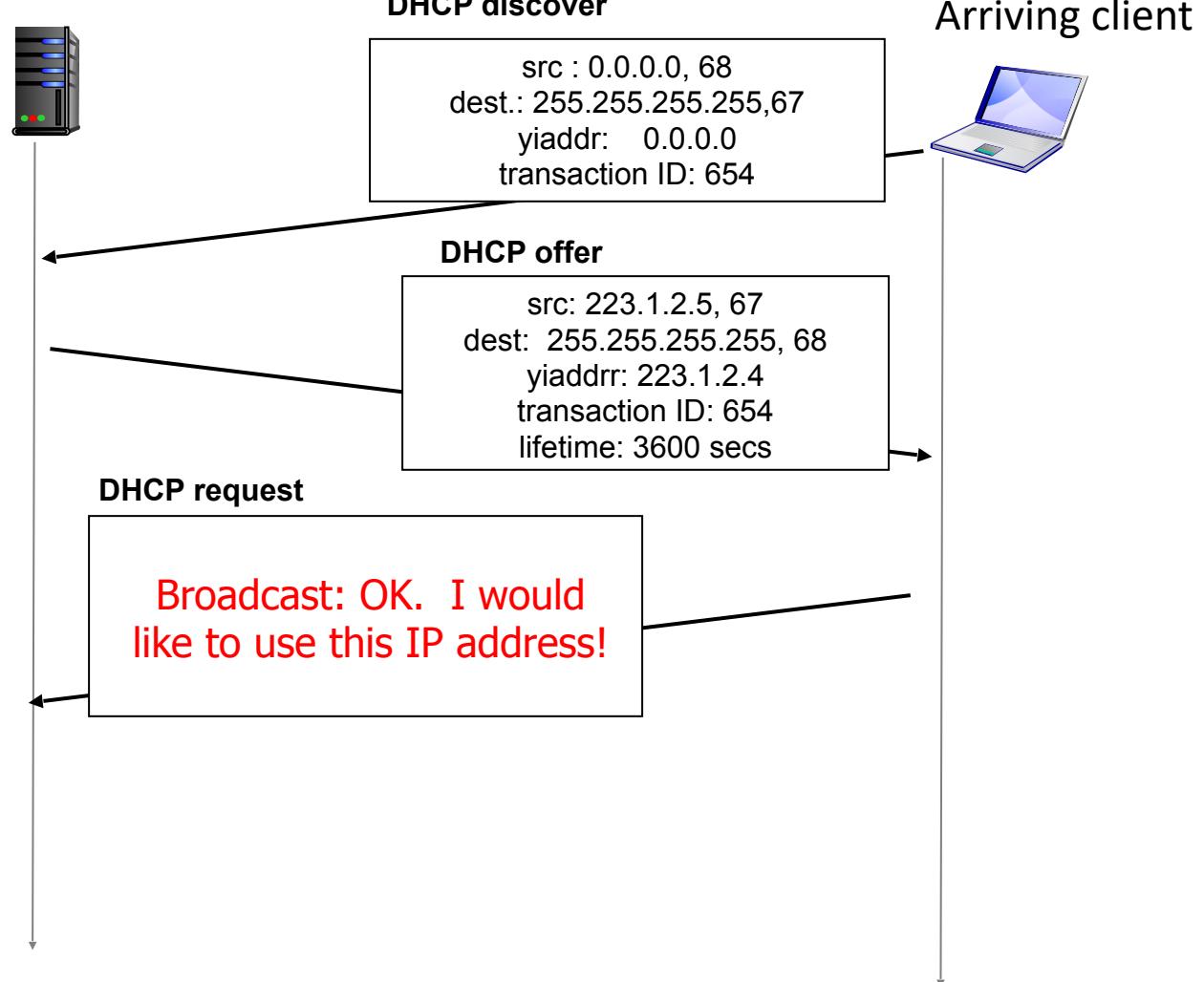
```
src: 223.1.2.5, 67  
dest: 255.255.255.255, 68  
yiaddr: 223.1.2.4  
transaction ID: 654  
lifetime: 3600 secs
```

The two steps above can be skipped “if a client remembers and wishes to reuse a previously allocated network address”
[RFC 2131]



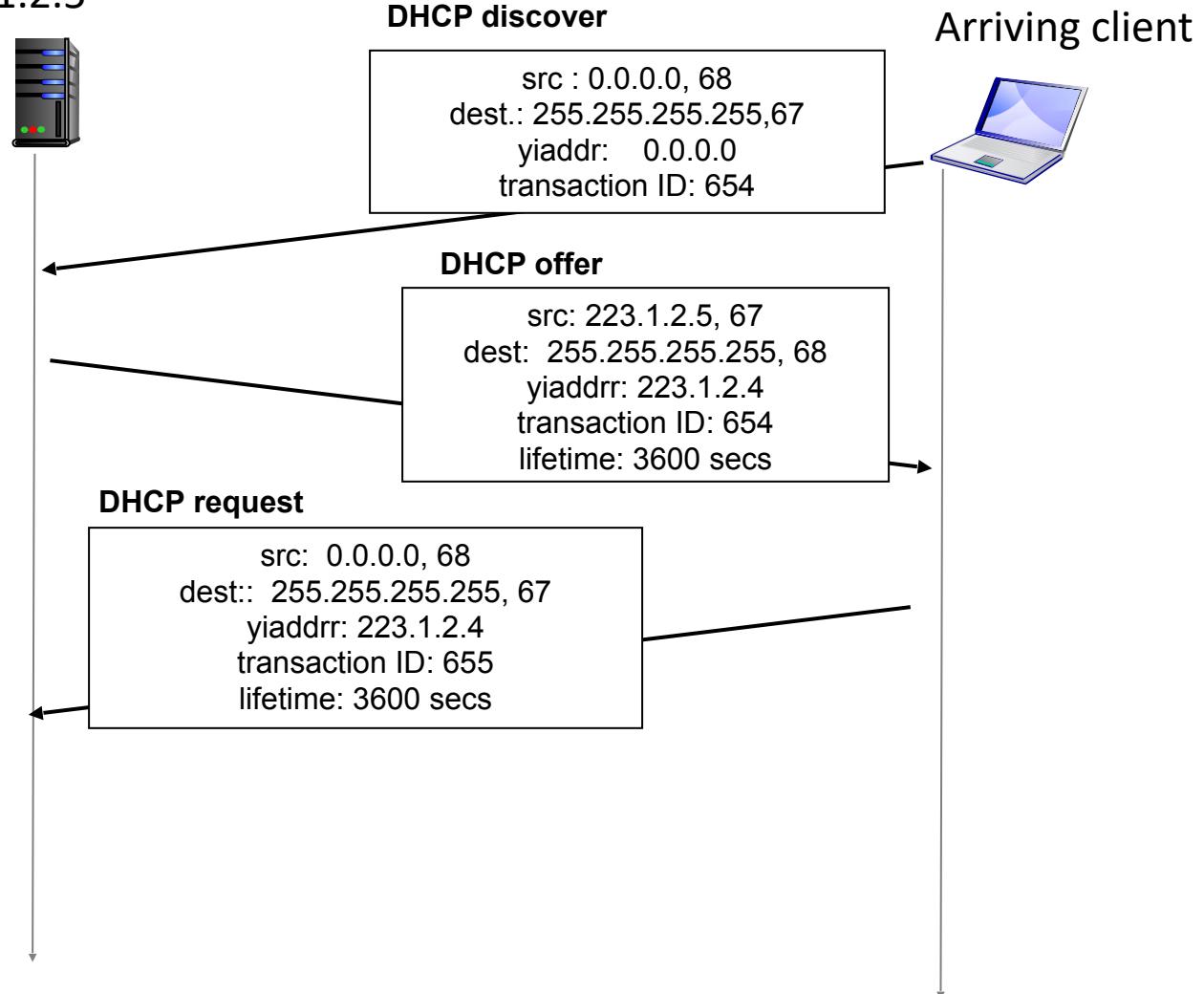
DHCP client-server scenario

DHCP server: 223.1.2.5

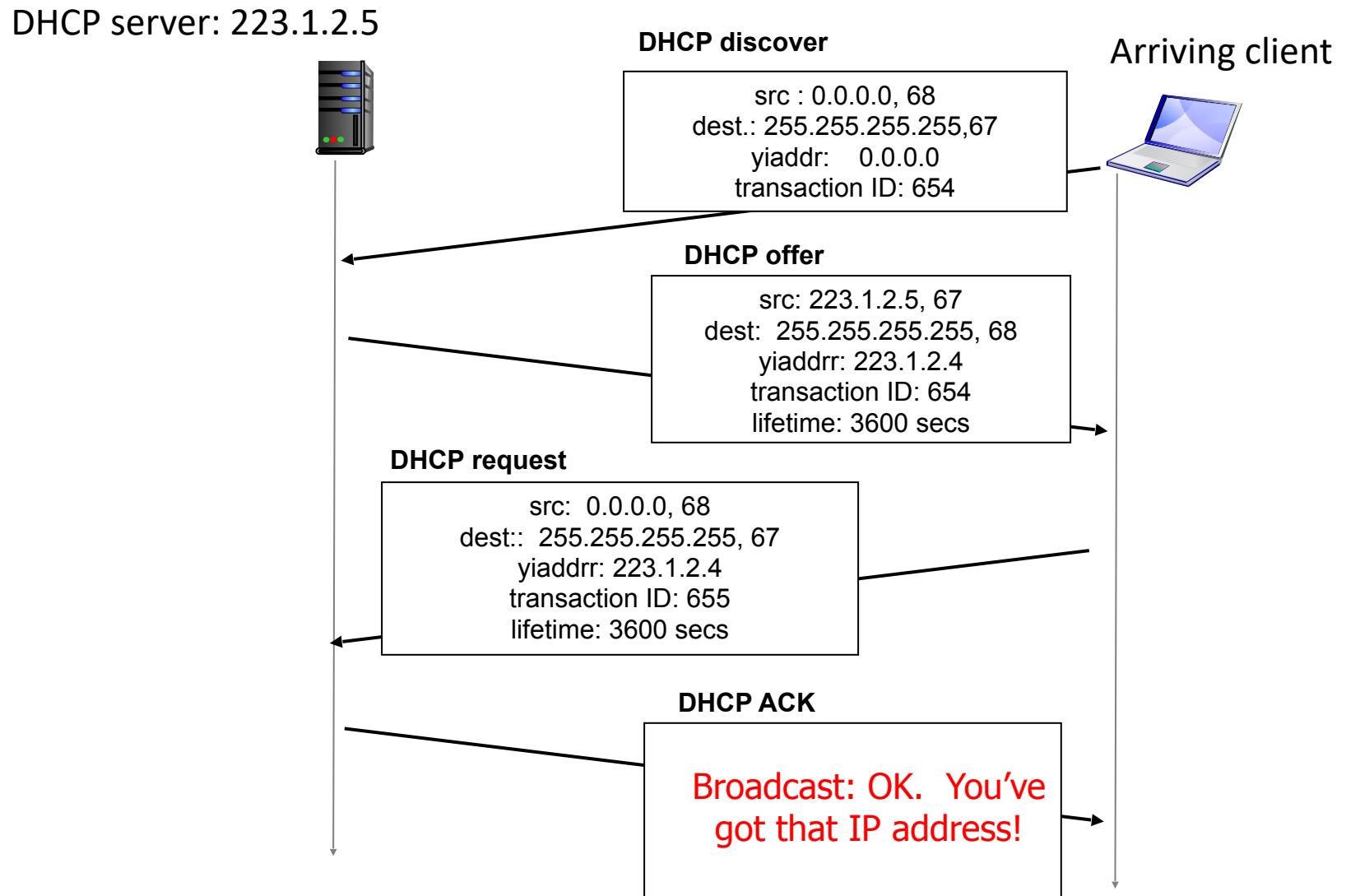


DHCP client-server scenario

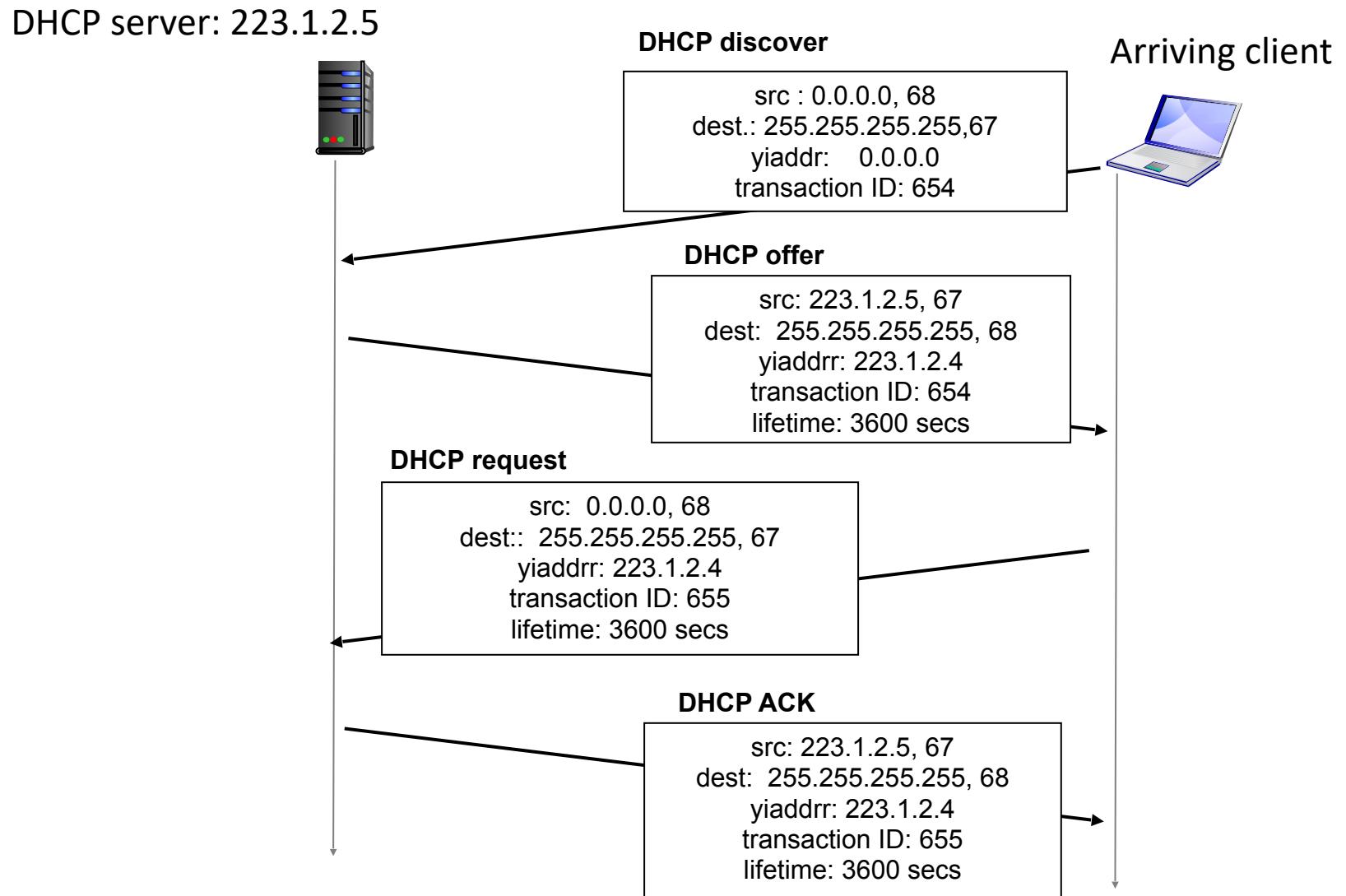
DHCP server: 223.1.2.5



DHCP client-server scenario



DHCP client-server scenario

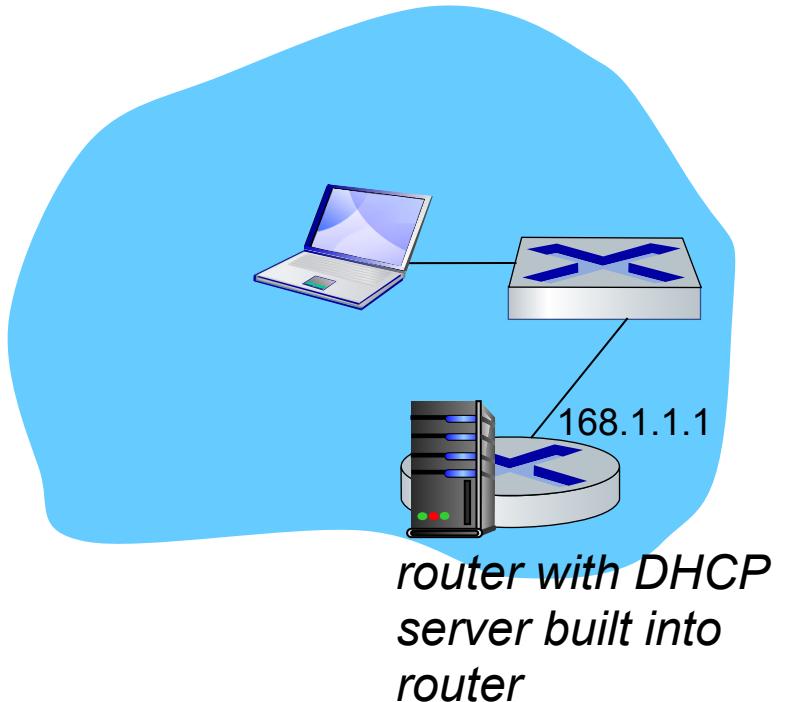


DHCP: more than IP addresses

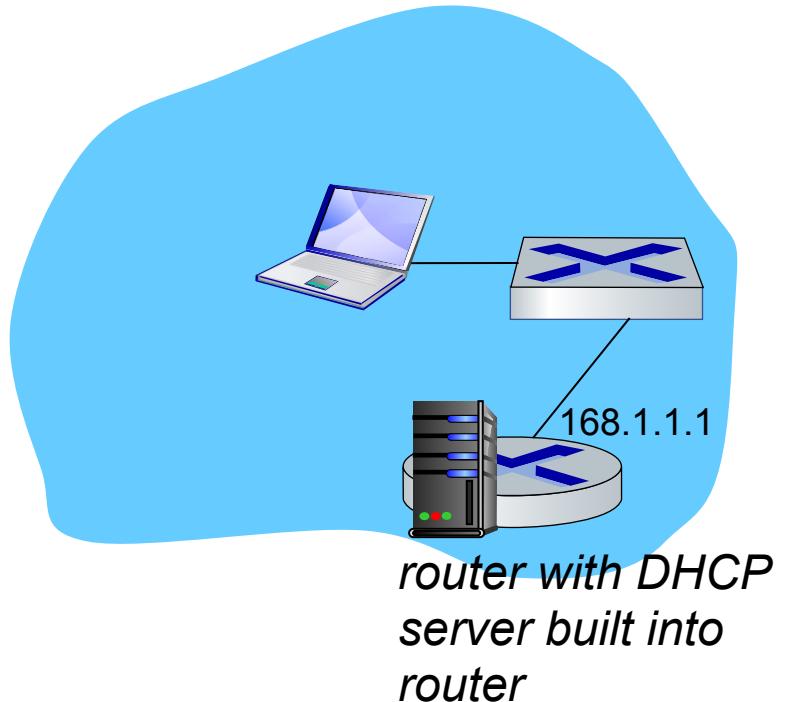
DHCP can return more than just allocated IP address on subnet:

- address of first-hop router for client
- name and IP address of DNS sever
- network mask (indicating network versus host portion of address)

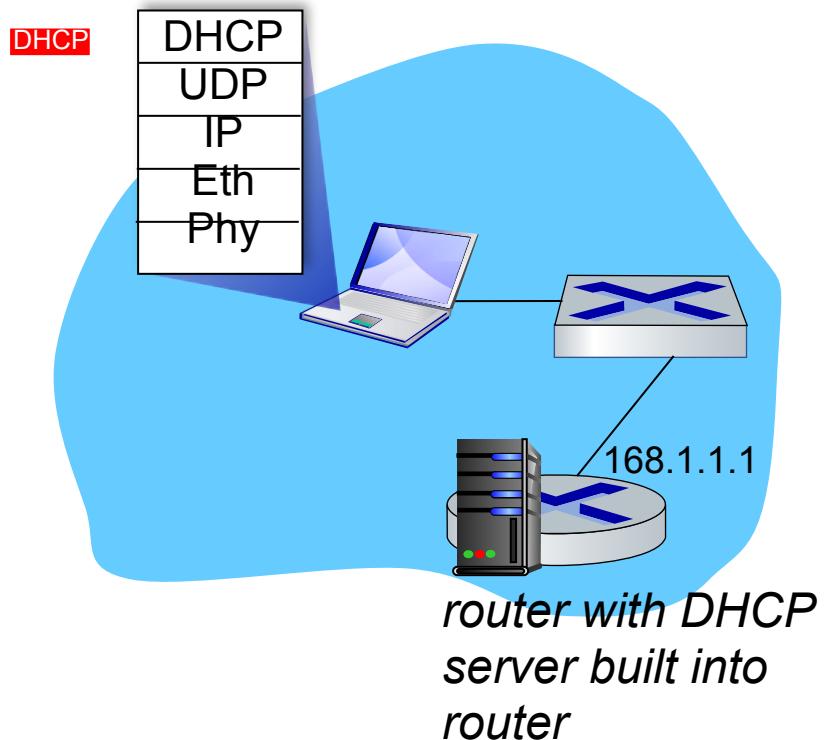
DHCP: example



DHCP: example

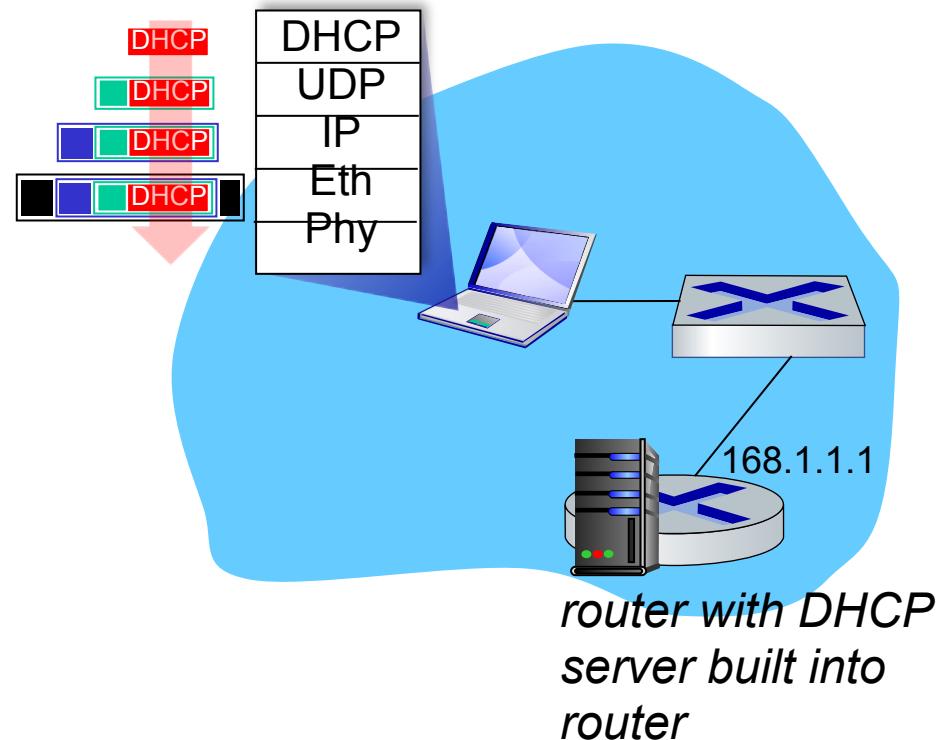


DHCP: example



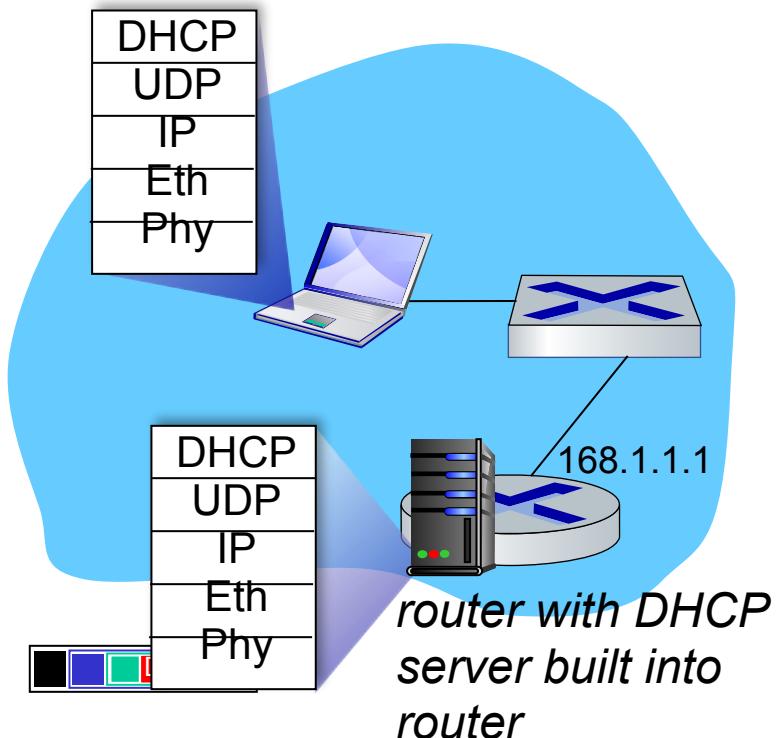
- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.

DHCP: example



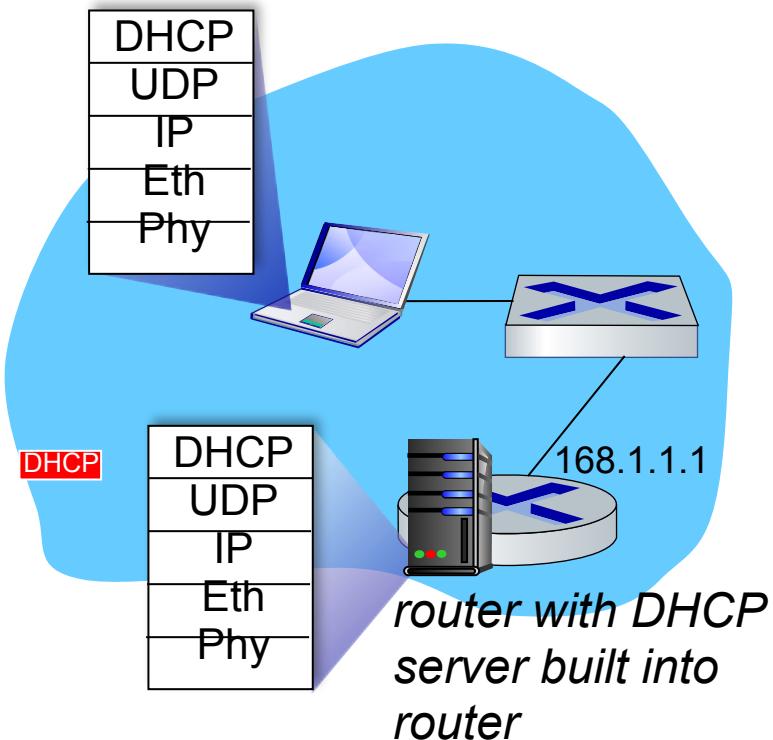
- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet

DHCP: example



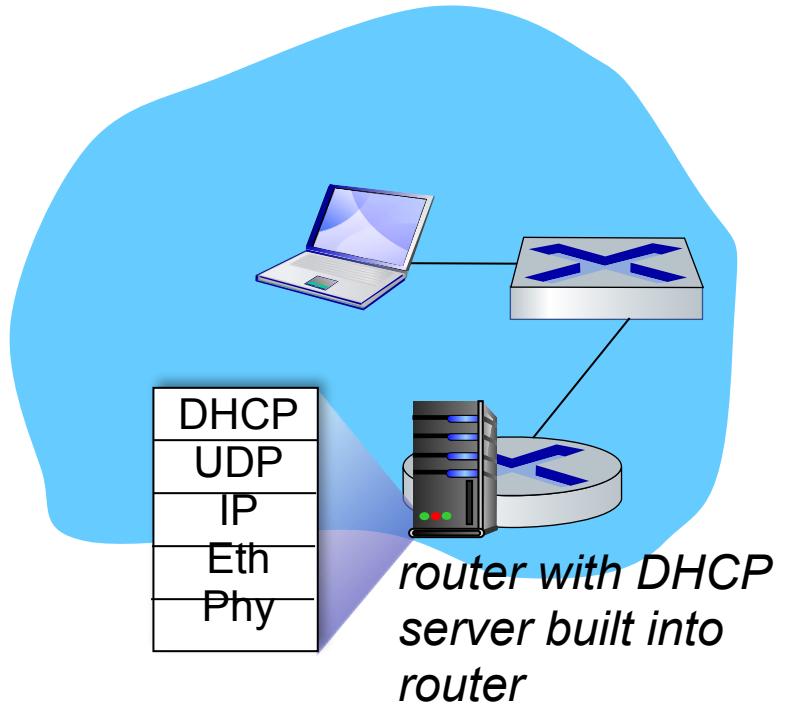
- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server

DHCP: example

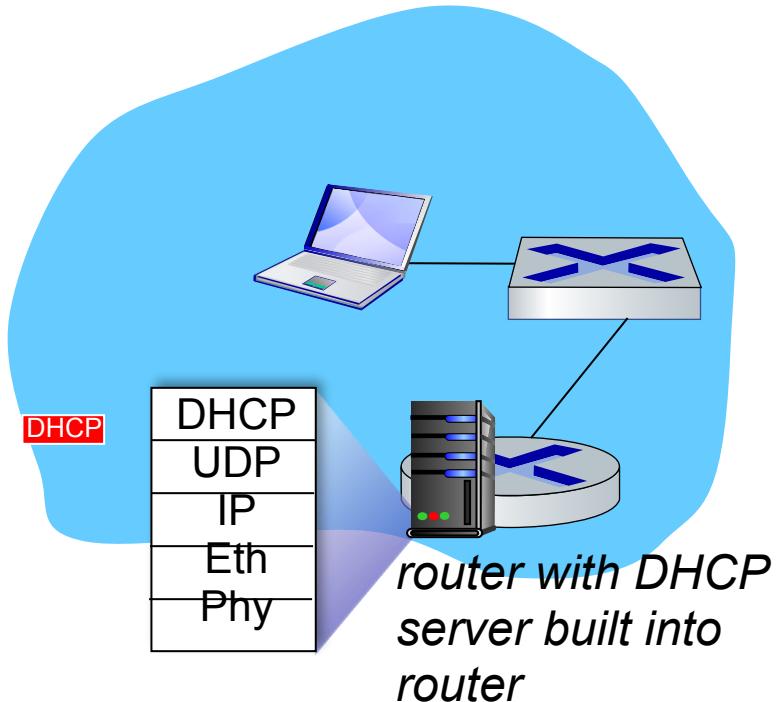


- Connecting laptop will use DHCP to get IP address, address of first-hop router, address of DNS server.
- DHCP REQUEST message encapsulated in UDP, encapsulated in IP, encapsulated in Ethernet
- Ethernet frame broadcast (dest: FFFFFFFFFFFF) on LAN, received at router running DHCP server
- Ethernet demux'ed to IP demux'ed, UDP demux'ed to DHCP

DHCP: example

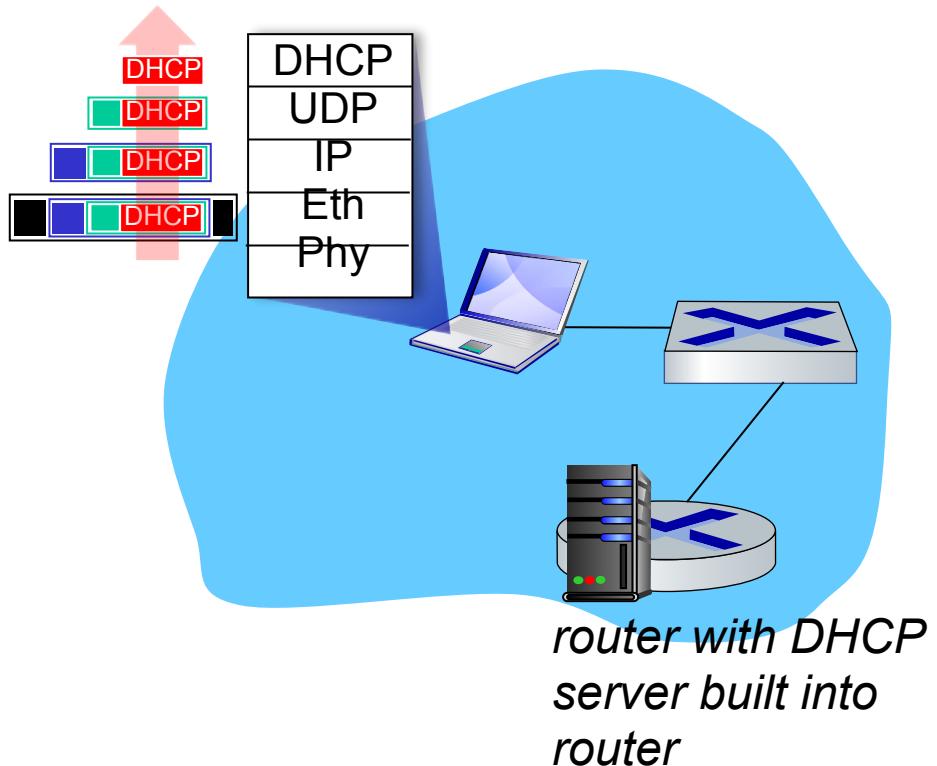


DHCP: example



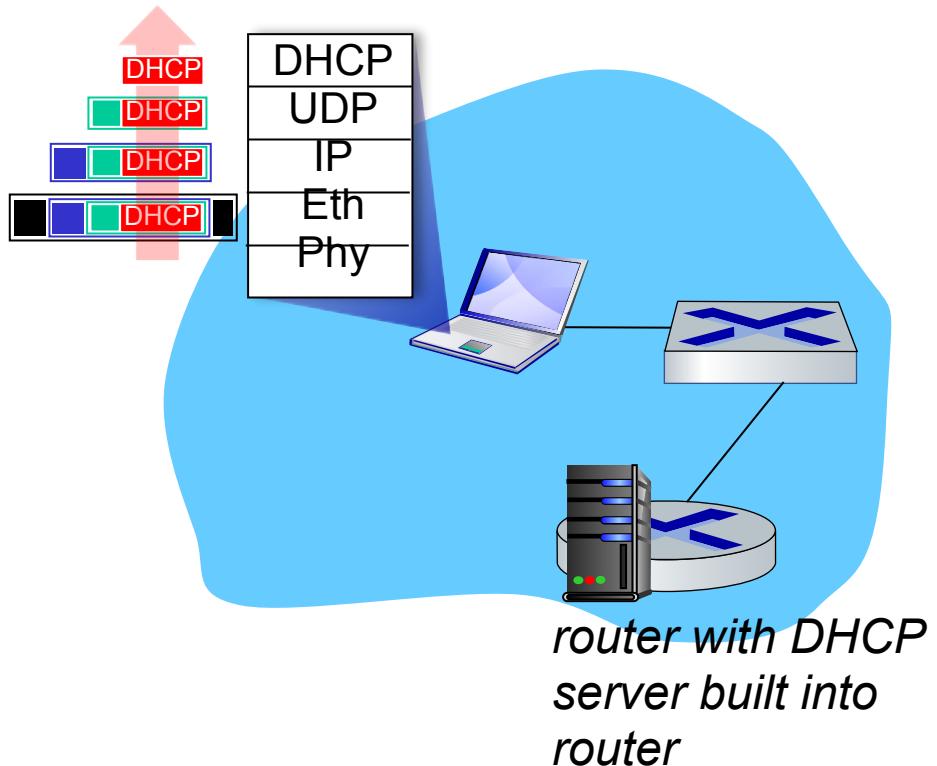
- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

DHCP: example



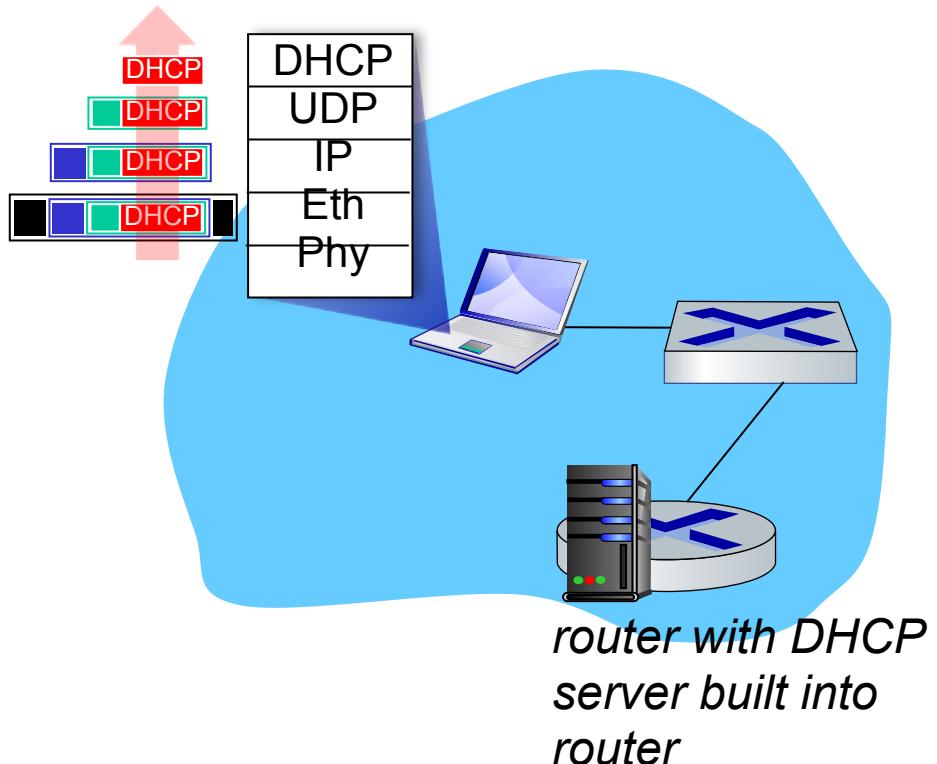
- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP: example



- DCP server formulates DHCP ACK containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server
- encapsulated DHCP server reply forwarded to client, demuxing up to DHCP at client
- client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

ISP's block

11001000 00010111 00010000 00000000 200.23.16.0/20

IP addresses: how to get one?

Q: how does *network* get subnet part of IP address?

A: gets allocated portion of its provider ISP's address space

ISP's block	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u> 200.23.16.0/20
-------------	--

ISP can then allocate out its address space in 8 blocks:

Organization 0	<u>11001000</u> <u>00010111</u> <u>00010000</u> <u>00000000</u> 200.23.16.0/23
----------------	--

Organization 1	<u>11001000</u> <u>00010111</u> <u>00010010</u> <u>00000000</u> 200.23.18.0/23
----------------	--

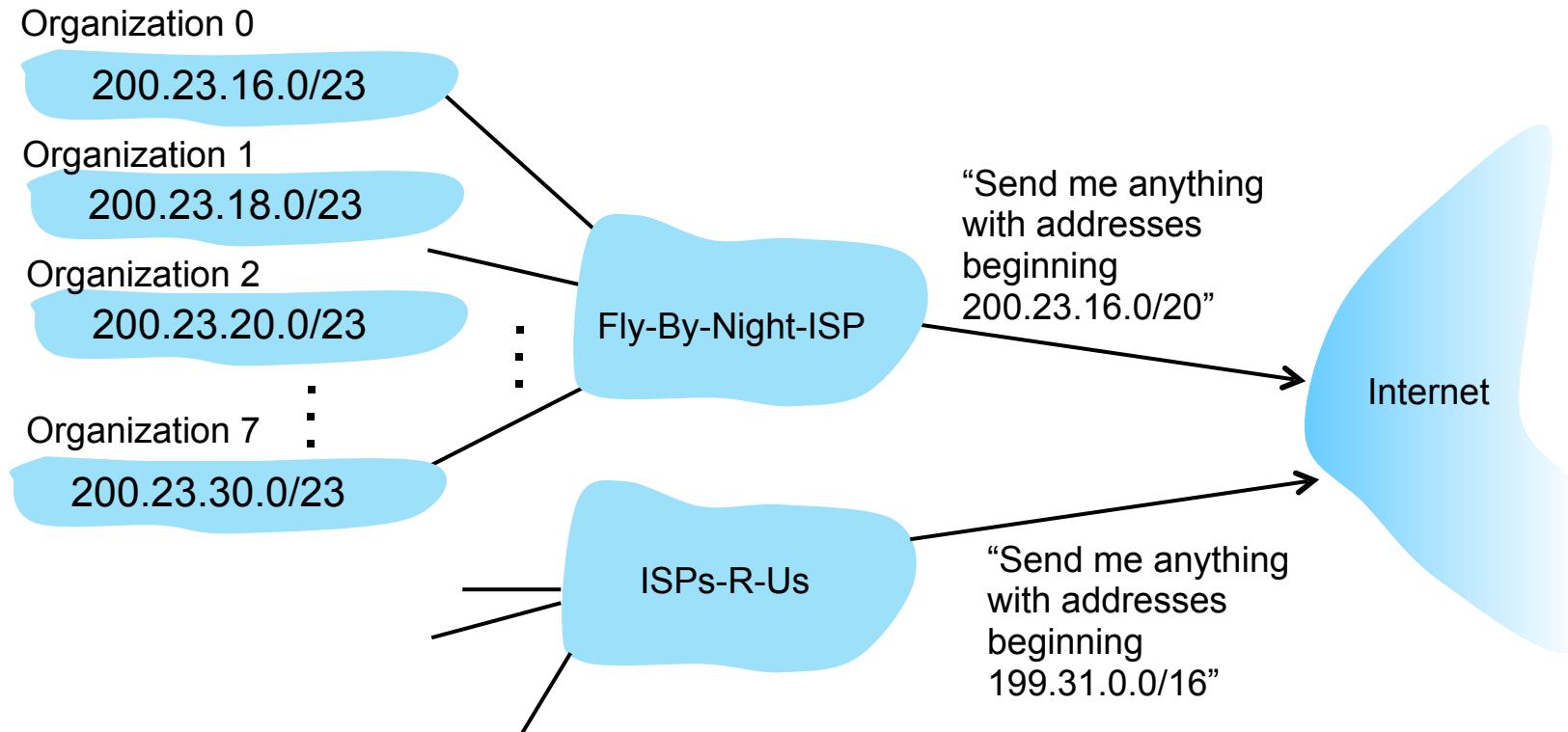
Organization 2	<u>11001000</u> <u>00010111</u> <u>00010100</u> <u>00000000</u> 200.23.20.0/23
----------------	--

...

Organization 7	<u>11001000</u> <u>00010111</u> <u>00011110</u> <u>00000000</u> 200.23.30.0/23
----------------	--

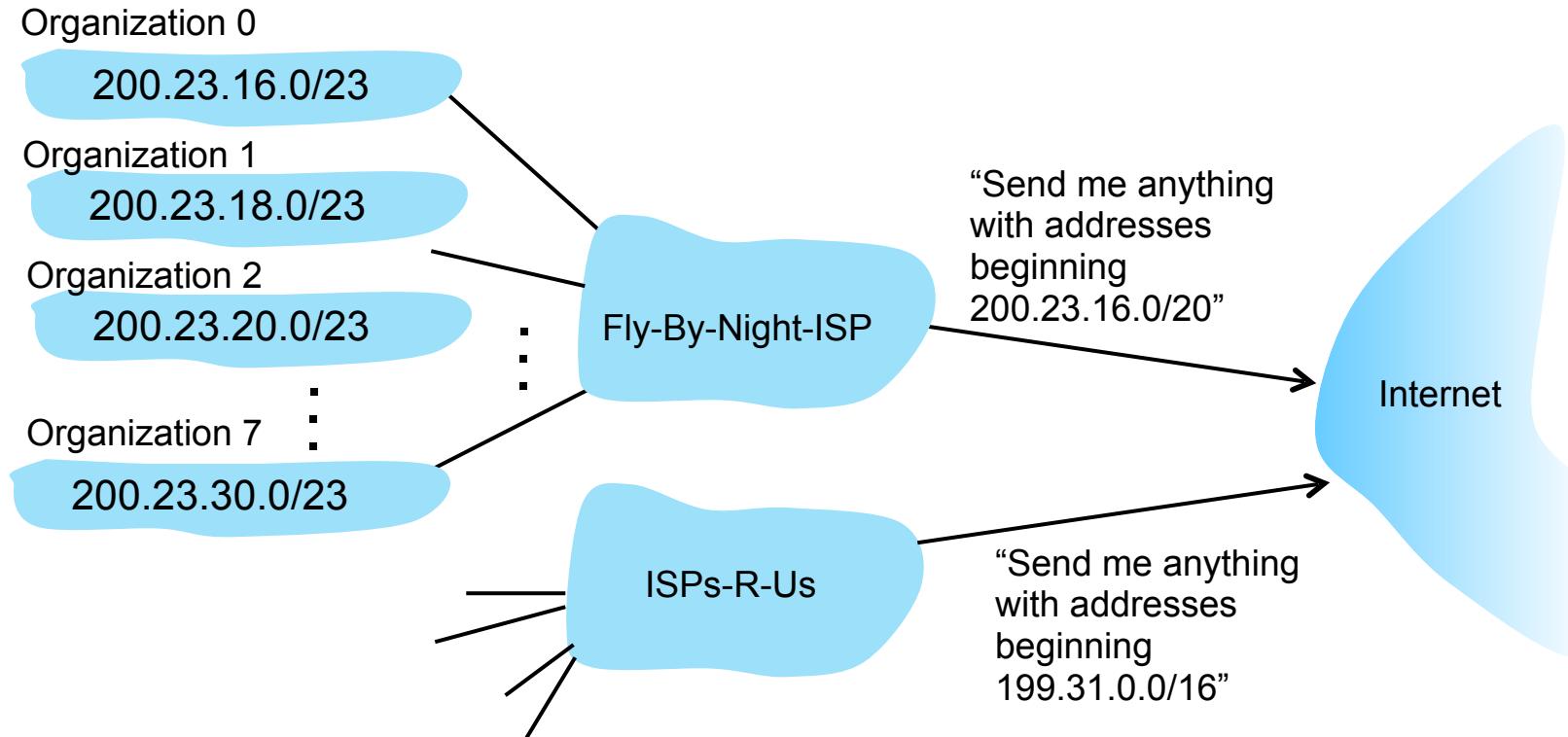
Hierarchical addressing: route aggregation

hierarchical addressing allows efficient advertisement of routing information:



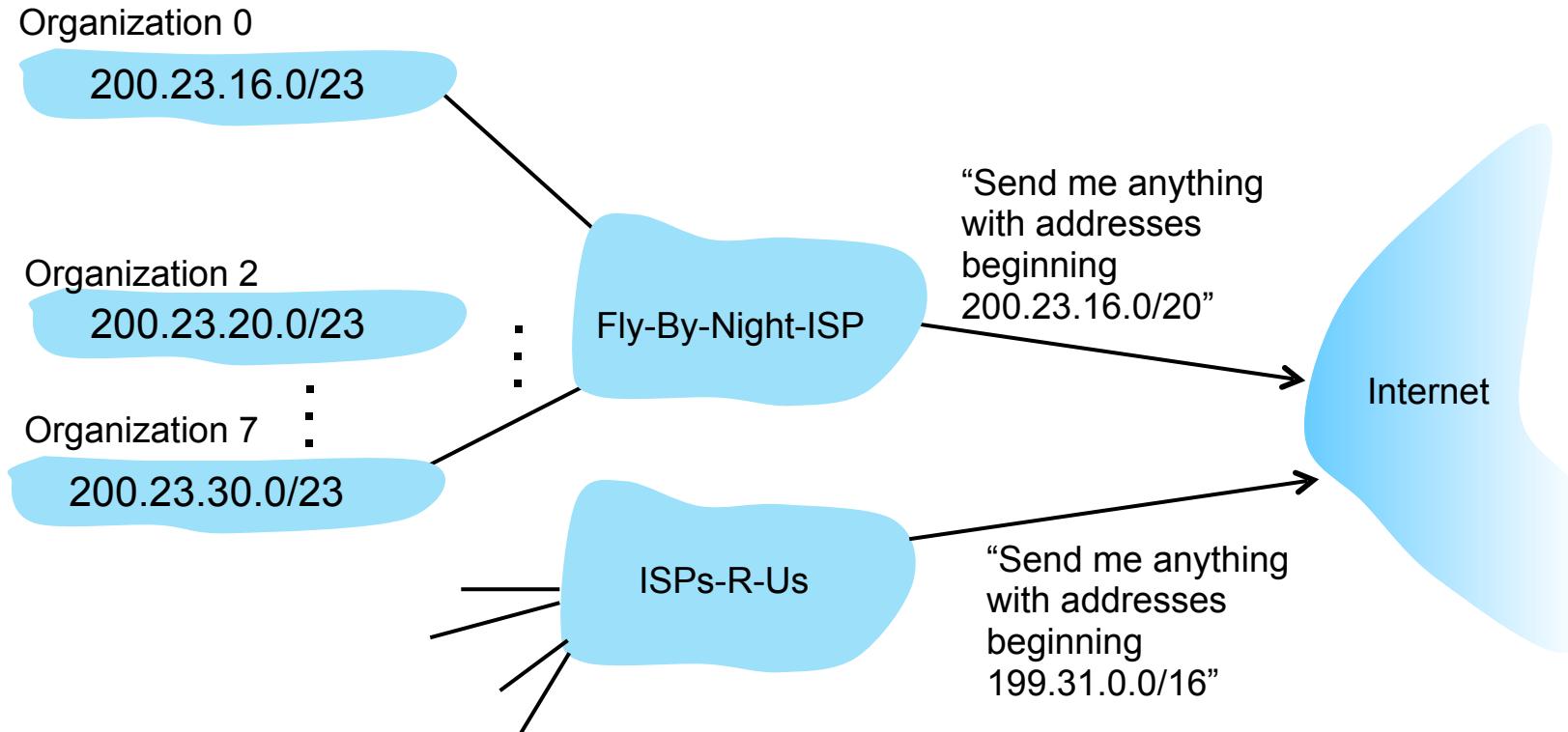
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



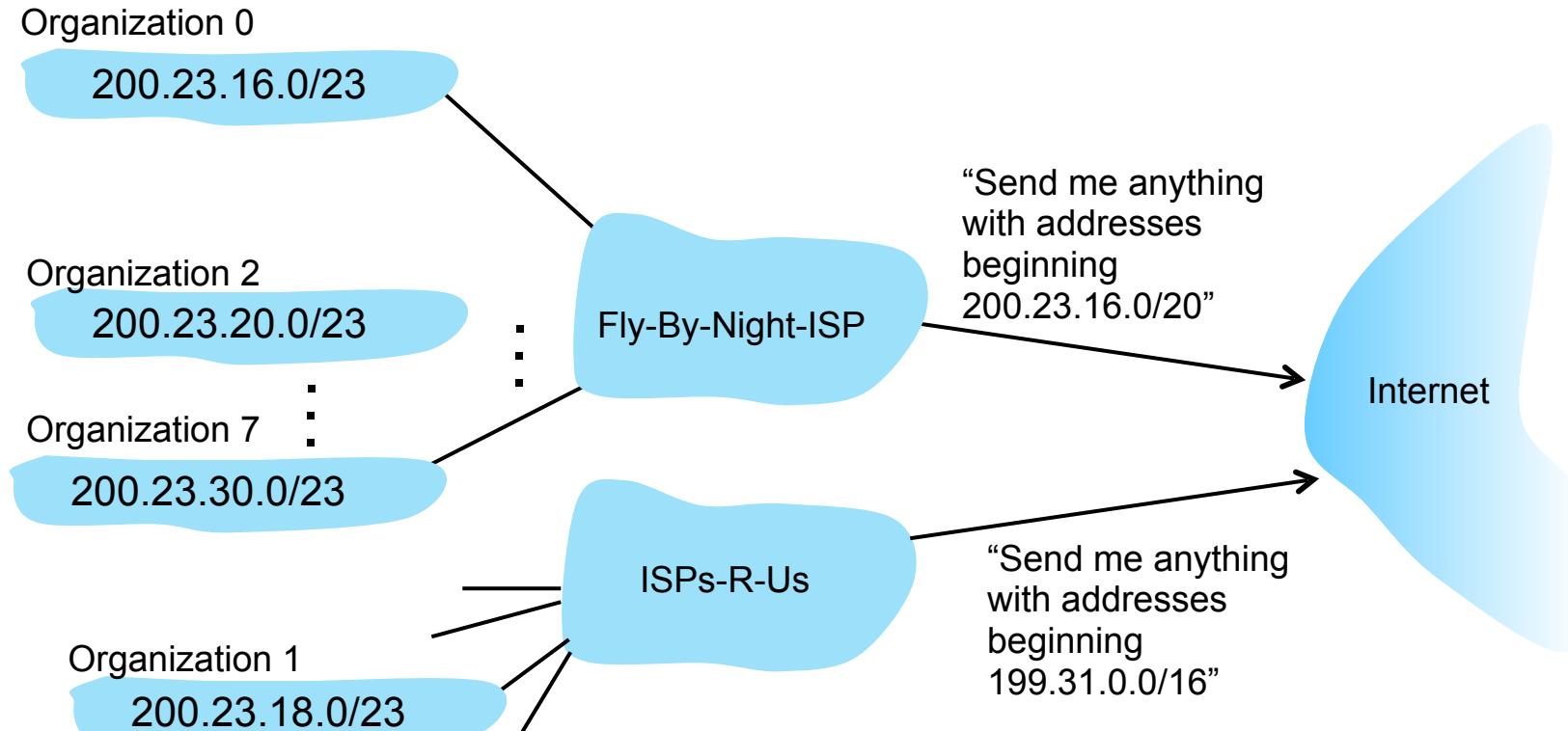
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



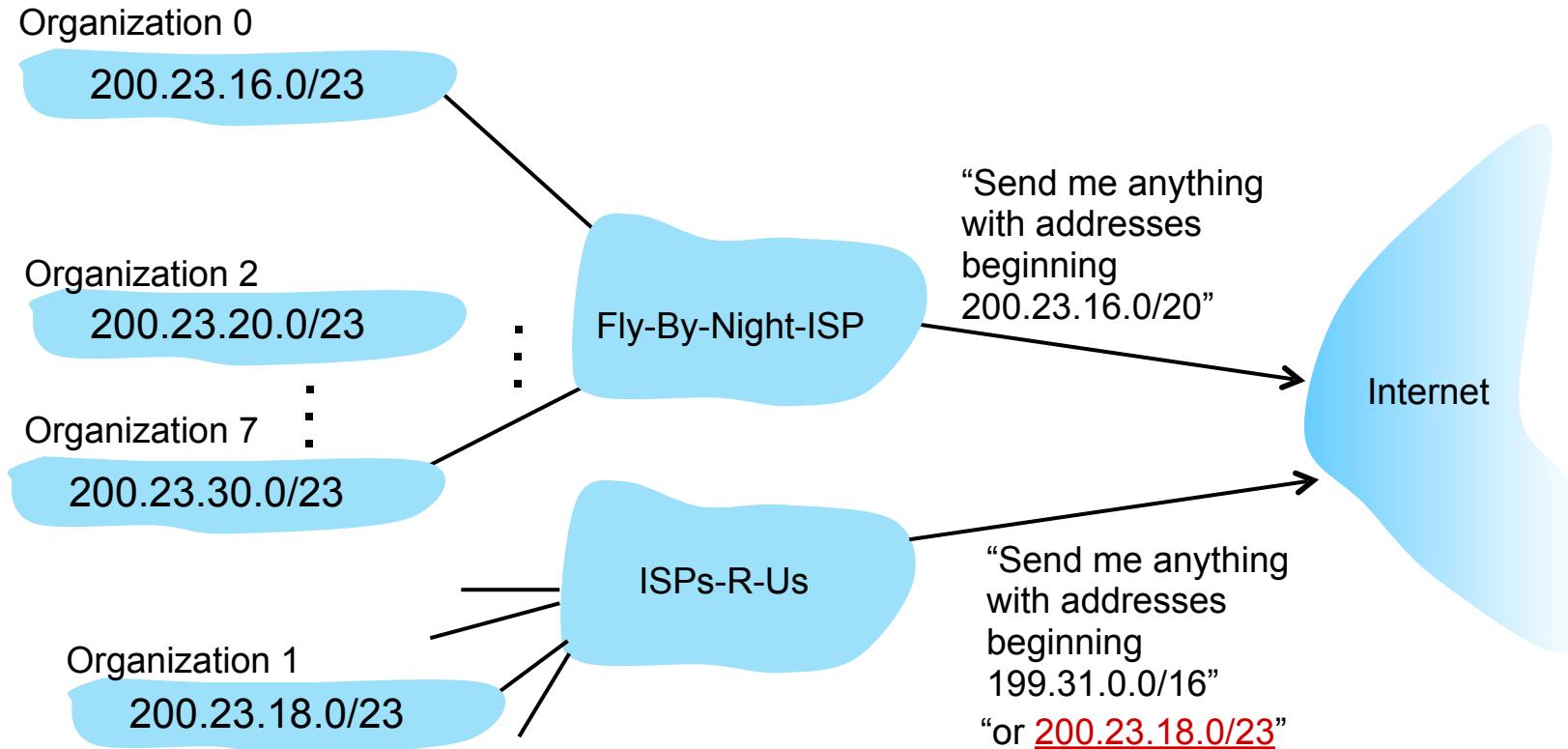
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



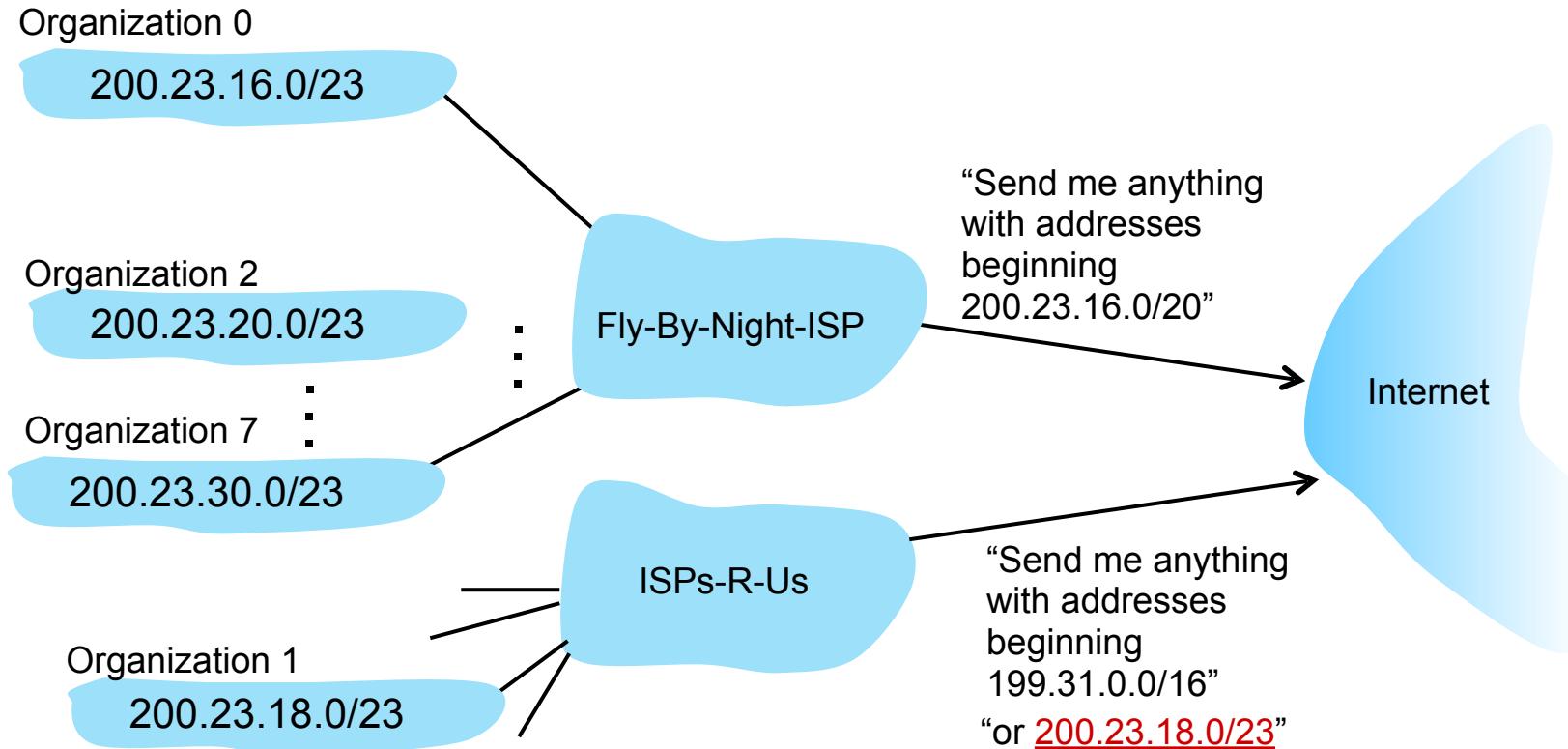
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



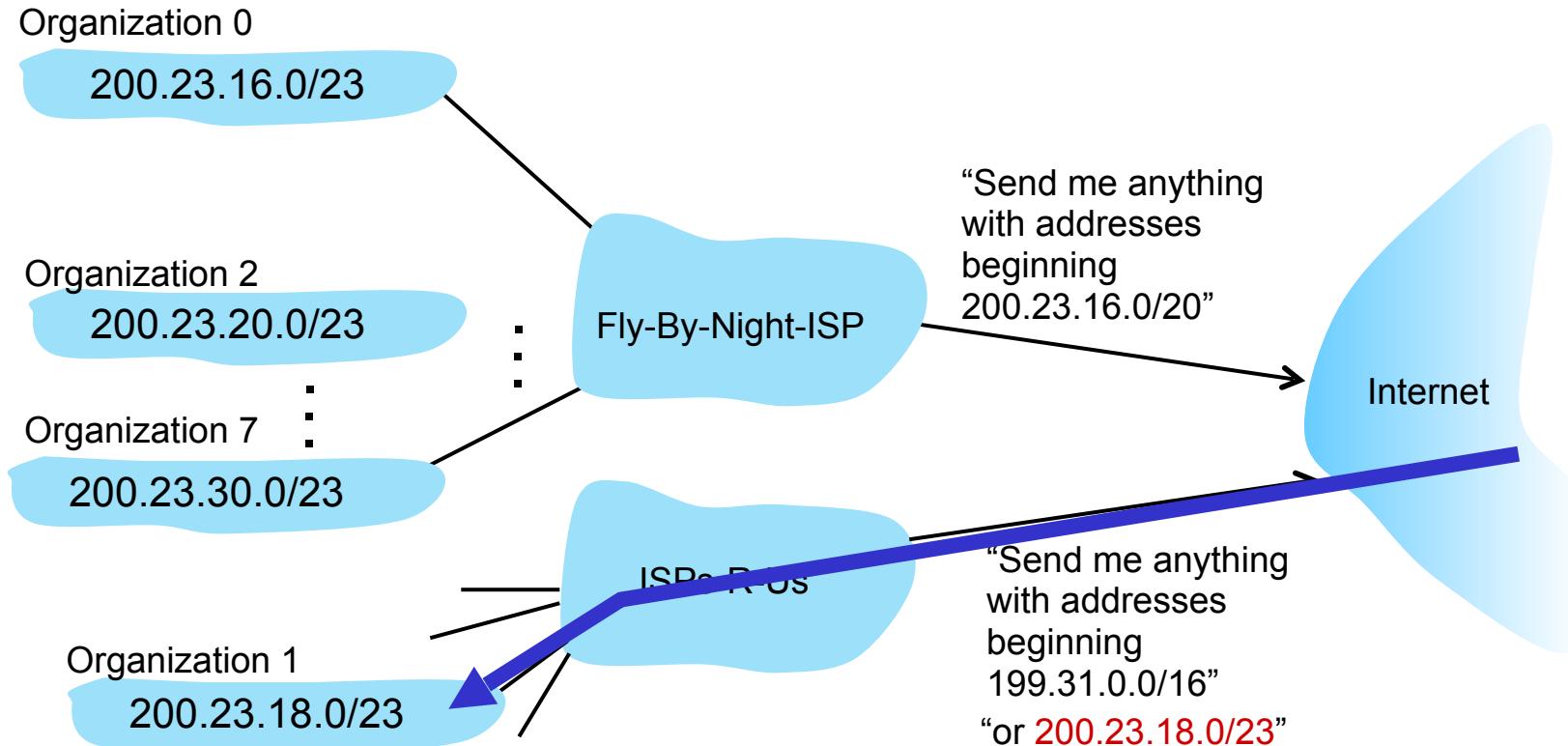
Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



Hierarchical addressing: more specific routes

- Organization 1 moves from Fly-By-Night-ISP to ISPs-R-Us
- ISPs-R-Us now advertises a more specific route to Organization 1



IP addressing: last words ...

IP addressing: last words ...

Q: how does an ISP get block of addresses?

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers
<http://www.icann.org/>

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

IP addressing: last words ...

Q: how does an ISP get block of addresses?

A: ICANN: Internet Corporation for Assigned Names and Numbers

<http://www.icann.org/>

- allocates IP addresses, through 5 regional registries (RRs) (who may then allocate to local registries)
- manages DNS root zone, including delegation of individual TLD (.com, .edu , ...) management

Q: are there enough 32-bit IP addresses?

- ICANN allocated last chunk of IPv4 addresses to RRs in 2011
- NAT (next) helps IPv4 address space exhaustion
- IPv6 has 128-bit address space

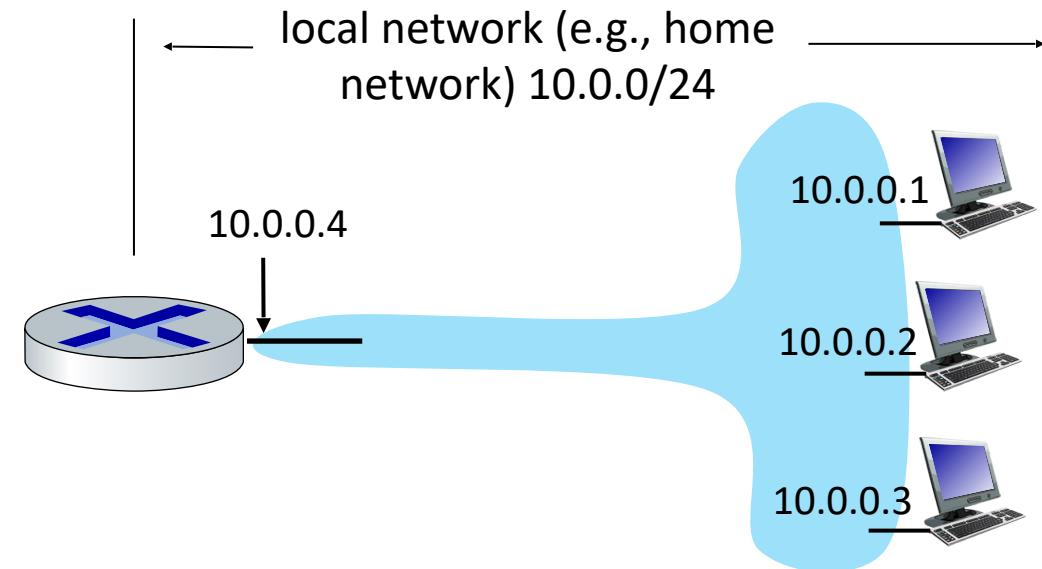
"Who the hell knew how much address space we needed?" Vint Cerf (reflecting on decision to make IPv4 address 32 bits long)

Chapter 4: roadmap

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - **network address translation**
 - fragmentation
 - ICMP
 - IPv6

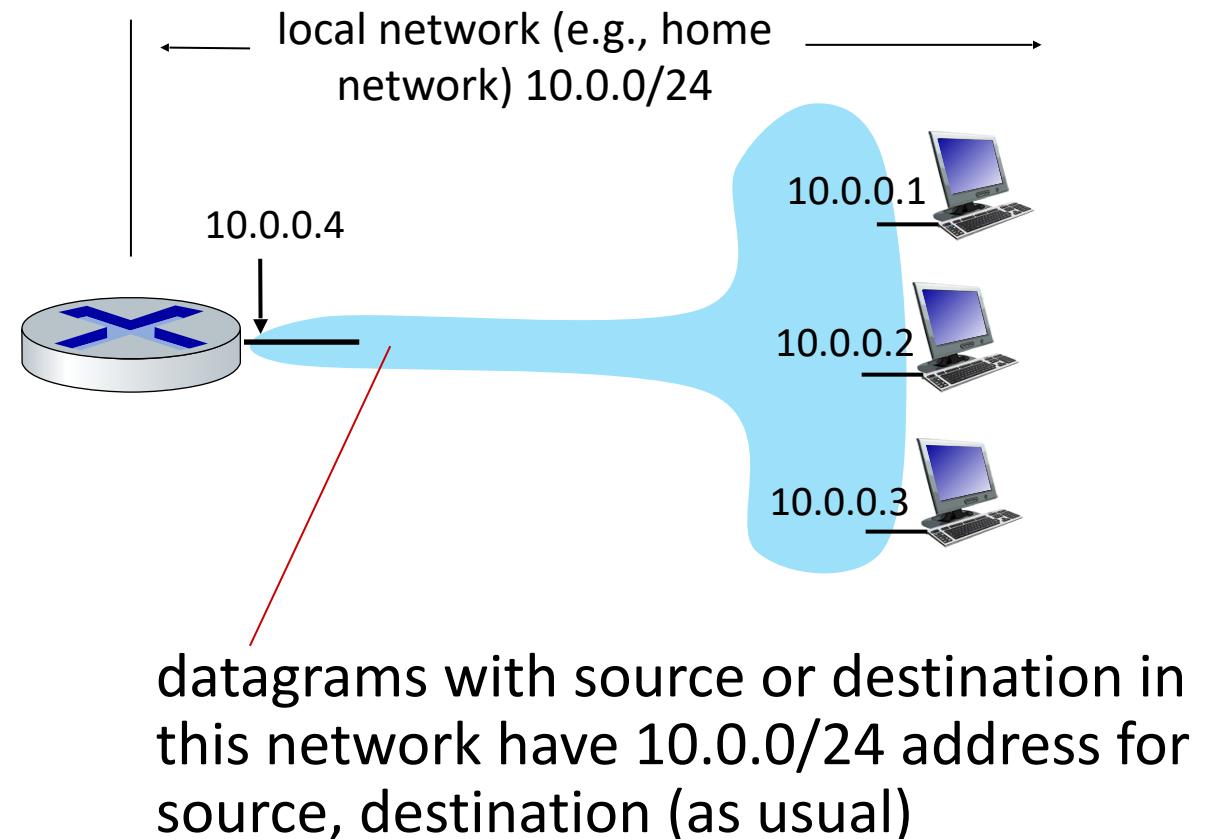
NAT: network address translation

NAT: all devices in local network share just **one** IPv4 address as far as outside world is concerned



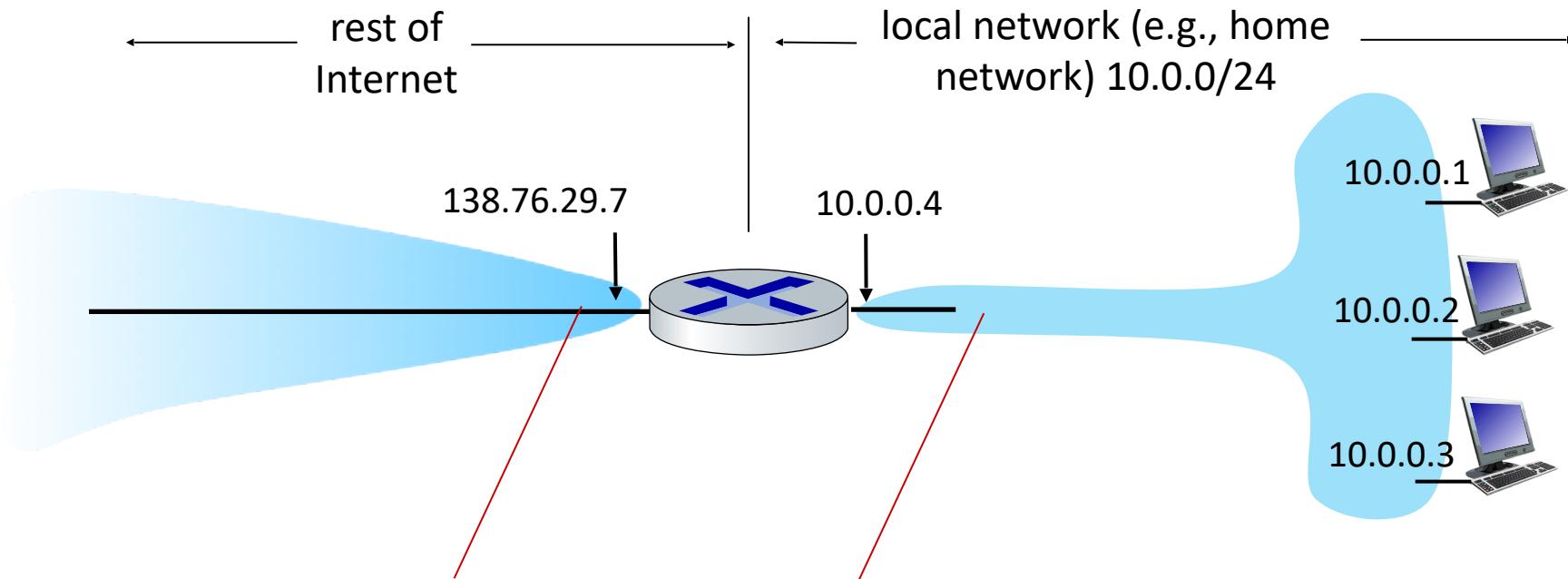
NAT: network address translation

NAT: all devices in local network share just **one** IPv4 address as far as outside world is concerned



NAT: network address translation

NAT: all devices in local network share just **one** IPv4 address as far as outside world is concerned



all datagrams *leaving* local network have *same* source NAT IP address: 138.76.29.7, but *different* source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

NAT: network address translation

NAT: network address translation

- all devices in local network have 32-bit addresses in a “private” IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network

NAT: network address translation

- all devices in local network have 32-bit addresses in a “private” IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
 - just **one** IP address needed from provider ISP for *all* devices
 - can change addresses of host in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - security: devices inside local net not directly addressable, visible by outside world

NAT: network address translation

- all devices in local network have 32-bit addresses in a “private” IP address space (10/8, 172.16/12, 192.168/16 prefixes) that can only be used in local network
- advantages:
 - just **one** IP address needed from provider ISP for *all* devices
 - can change addresses of host in local network without notifying outside world
 - can change ISP without changing addresses of devices in local network
 - security: devices inside local net not directly addressable, visible by outside world

NAT: network address translation

implementation: NAT router must (transparently):

NAT: network address translation

implementation: NAT router must (transparently):

- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 - remote clients/servers will respond using (NAT IP address, new port #) as destination address

NAT: network address translation

implementation: NAT router must (transparently):

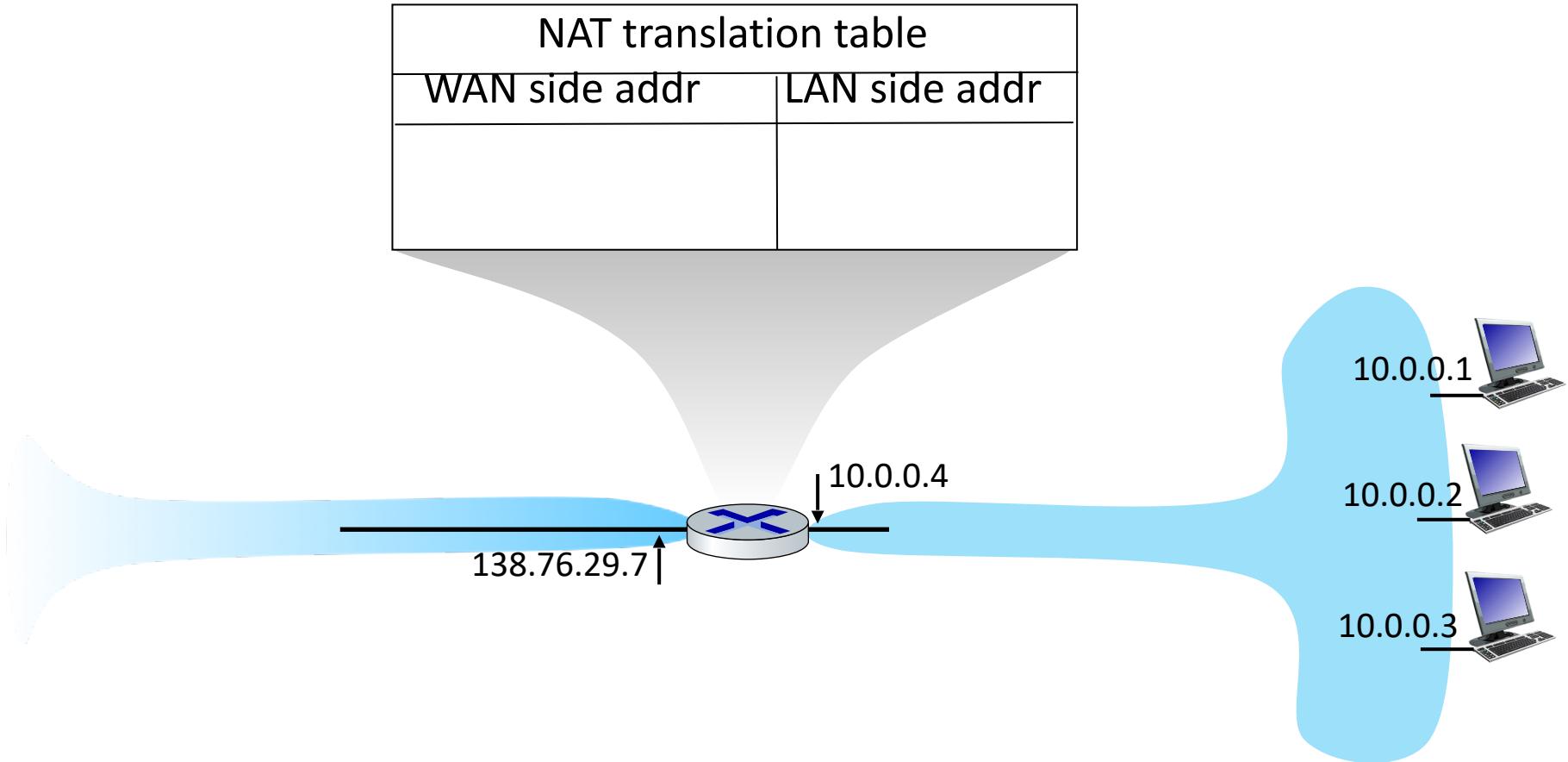
- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair

NAT: network address translation

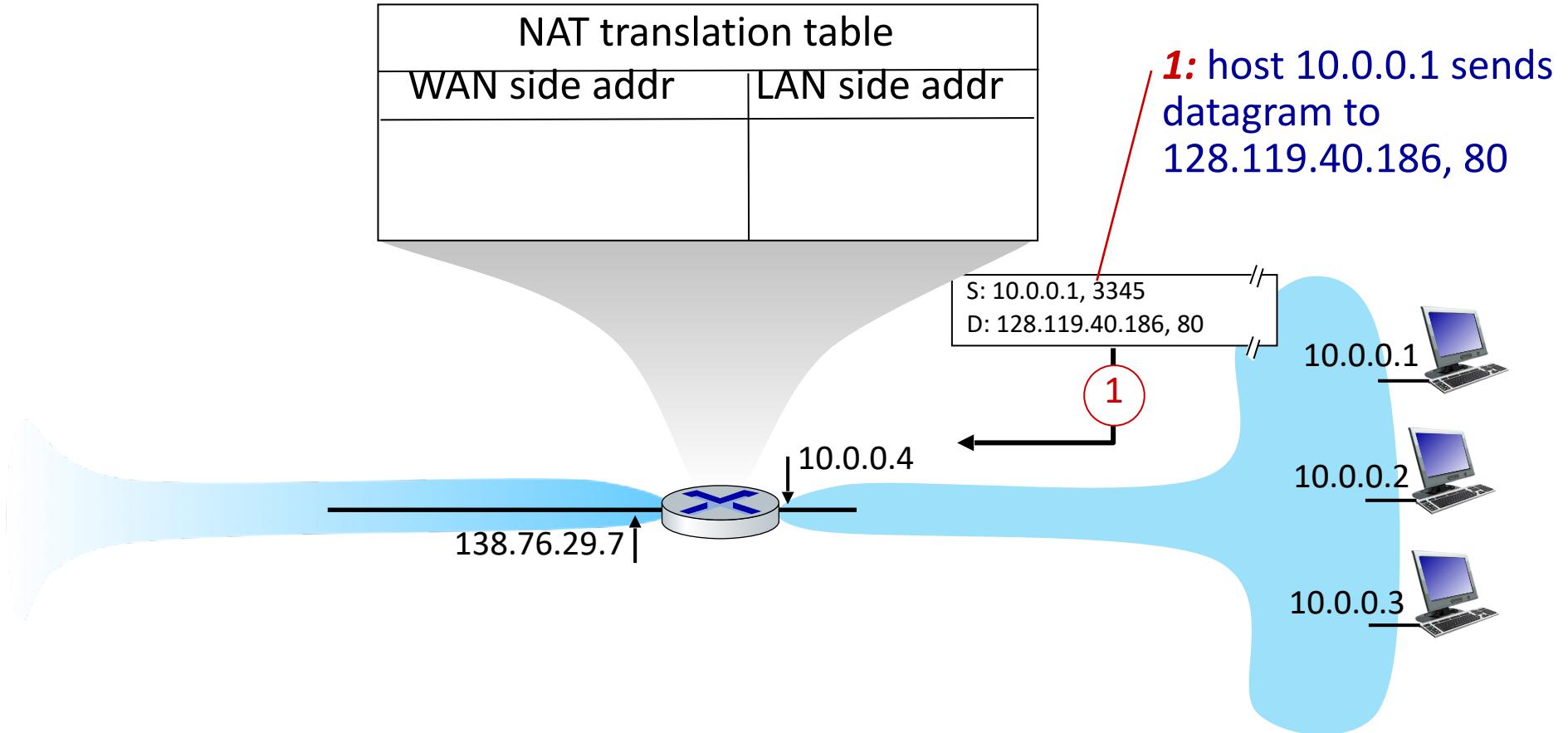
implementation: NAT router must (transparently):

- outgoing datagrams: replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
 - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
- incoming datagrams: replace (NAT IP address, new port #) in destination fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: network address translation



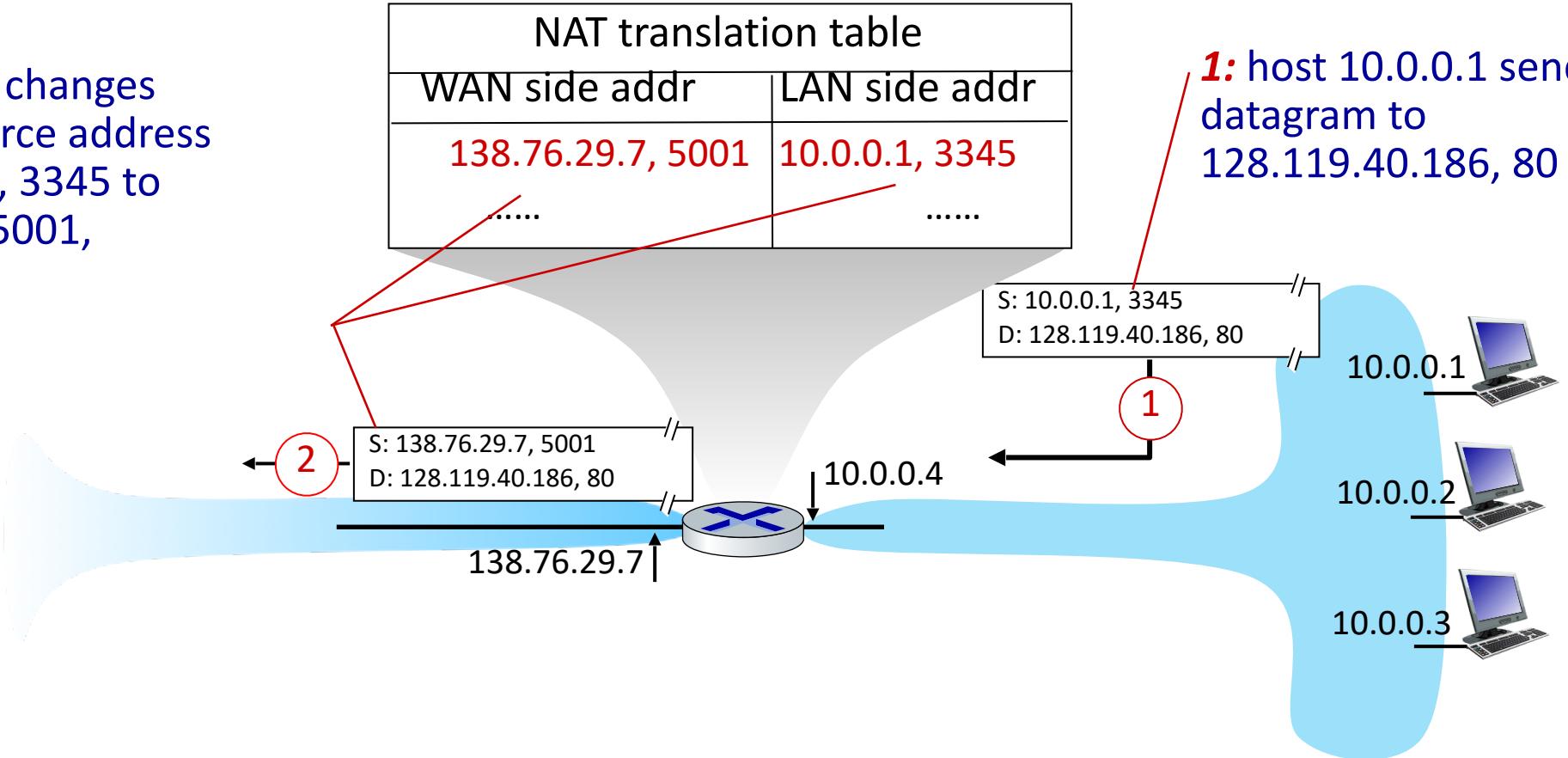
NAT: network address translation



NAT: network address translation

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80

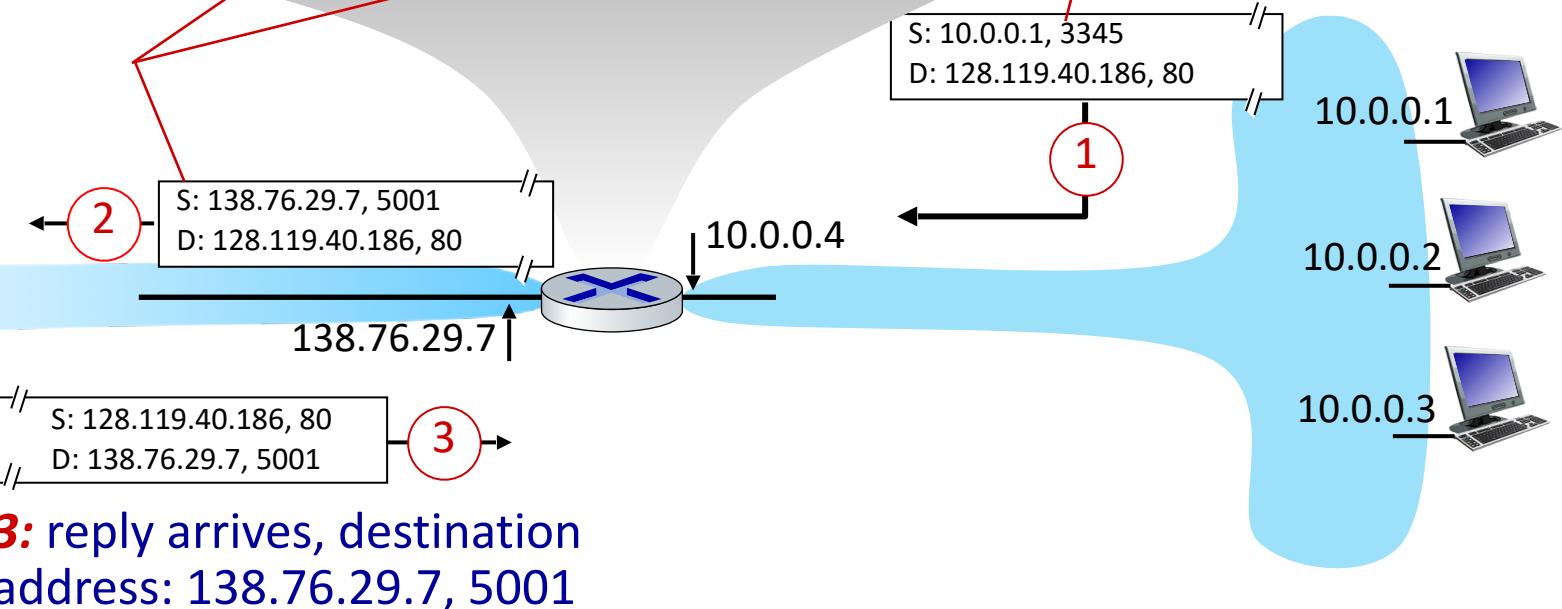


NAT: network address translation

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



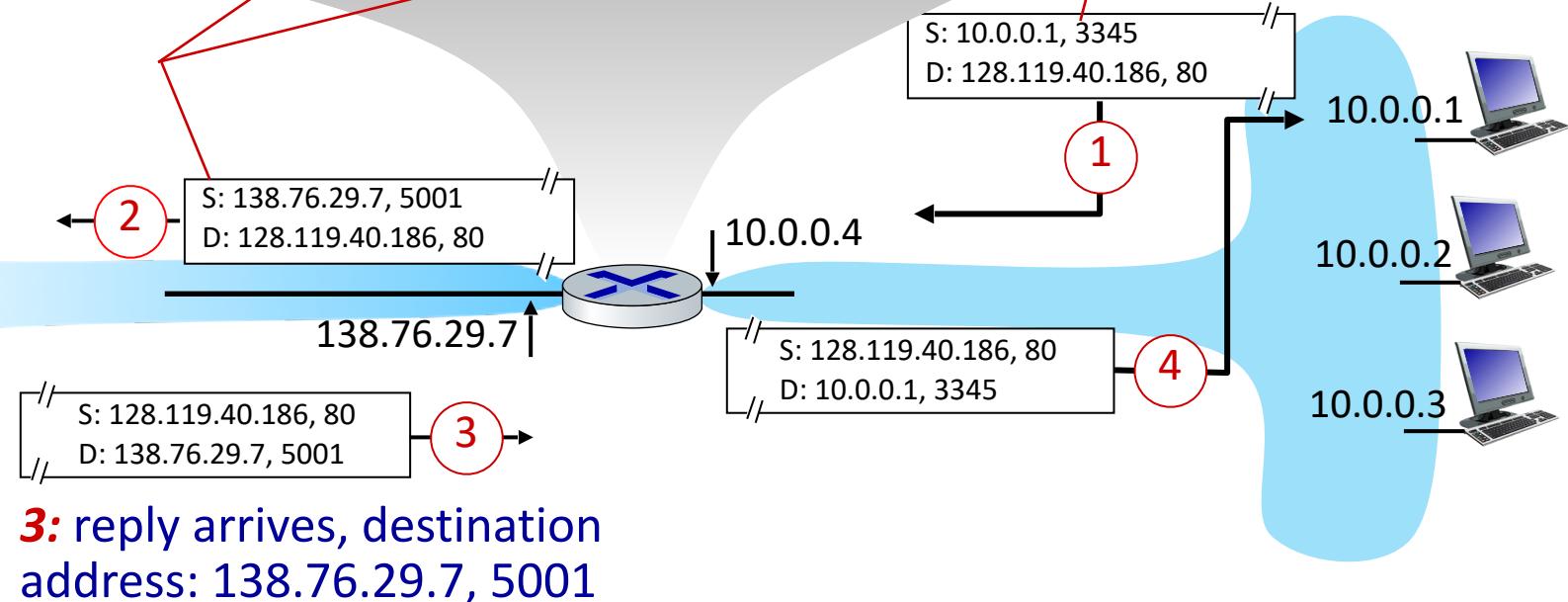
3: reply arrives, destination address: 138.76.29.7, 5001

NAT: network address translation

2: NAT router changes datagram source address from 10.0.0.1, 3345 to 138.76.29.7, 5001, updates table

NAT translation table	
WAN side addr	LAN side addr
138.76.29.7, 5001	10.0.0.1, 3345
.....

1: host 10.0.0.1 sends datagram to 128.119.40.186, 80



3: reply arrives, destination address: 138.76.29.7, 5001

NAT: network address translation

NAT: network address translation

- NAT has been controversial:
 - routers “should” only process up to layer 3
 - address “shortage” should be solved by IPv6
 - violates end-to-end argument (port # manipulation by network-layer device)
 - NAT traversal: what if client wants to connect to server behind NAT?

NAT: network address translation

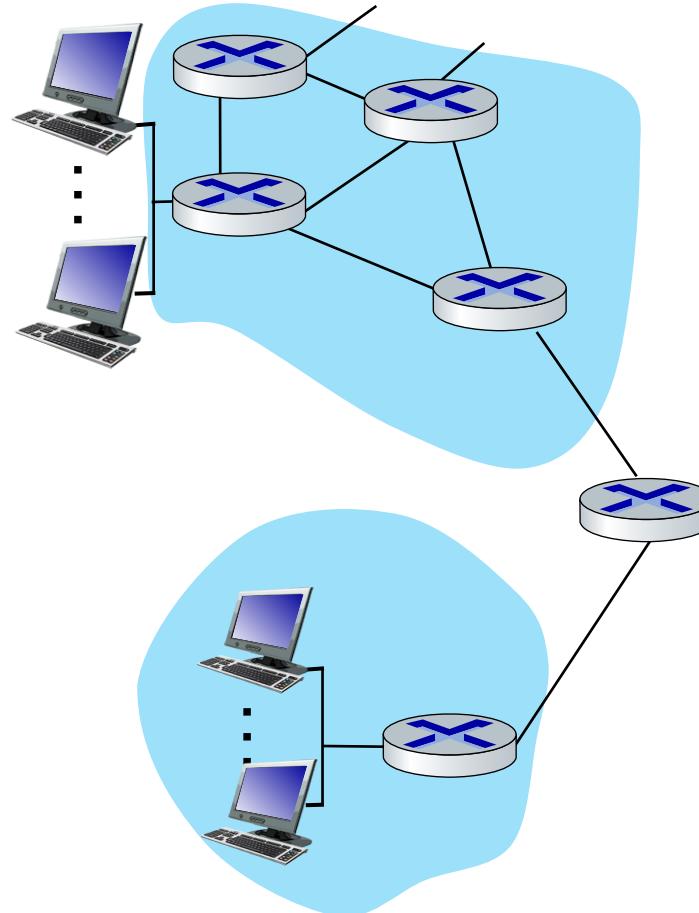
- NAT has been controversial:
 - routers “should” only process up to layer 3
 - address “shortage” should be solved by IPv6
 - violates end-to-end argument (port # manipulation by network-layer device)
 - NAT traversal: what if client wants to connect to server behind NAT?
- but NAT is here to stay:
 - extensively used in home and institutional nets, 4G/5G cellular nets

Chapter 4: roadmap

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - **fragmentation**
 - ICMP
 - IPv6

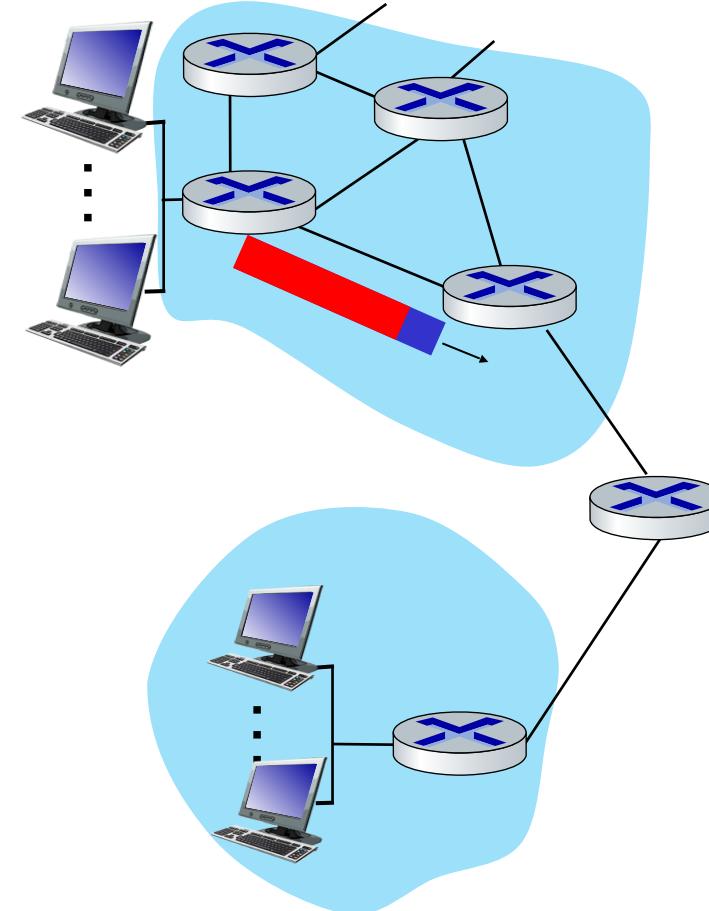
IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



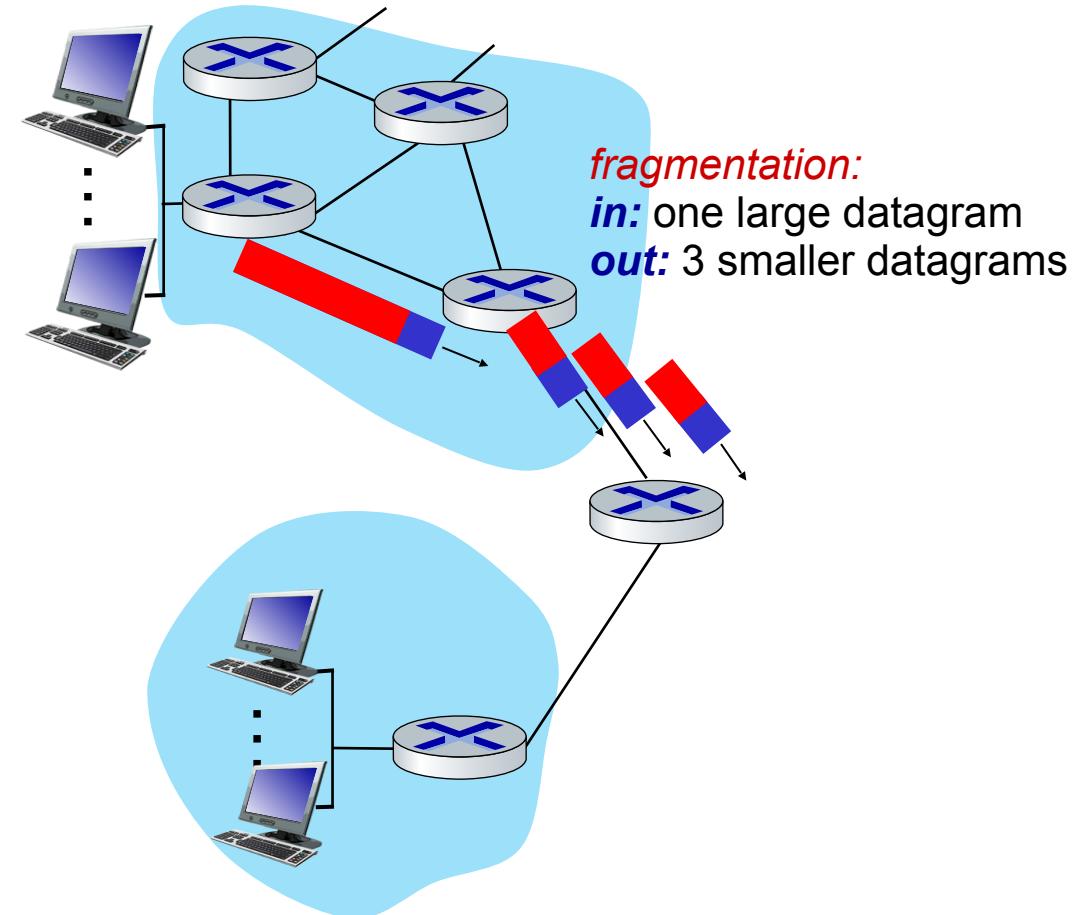
IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



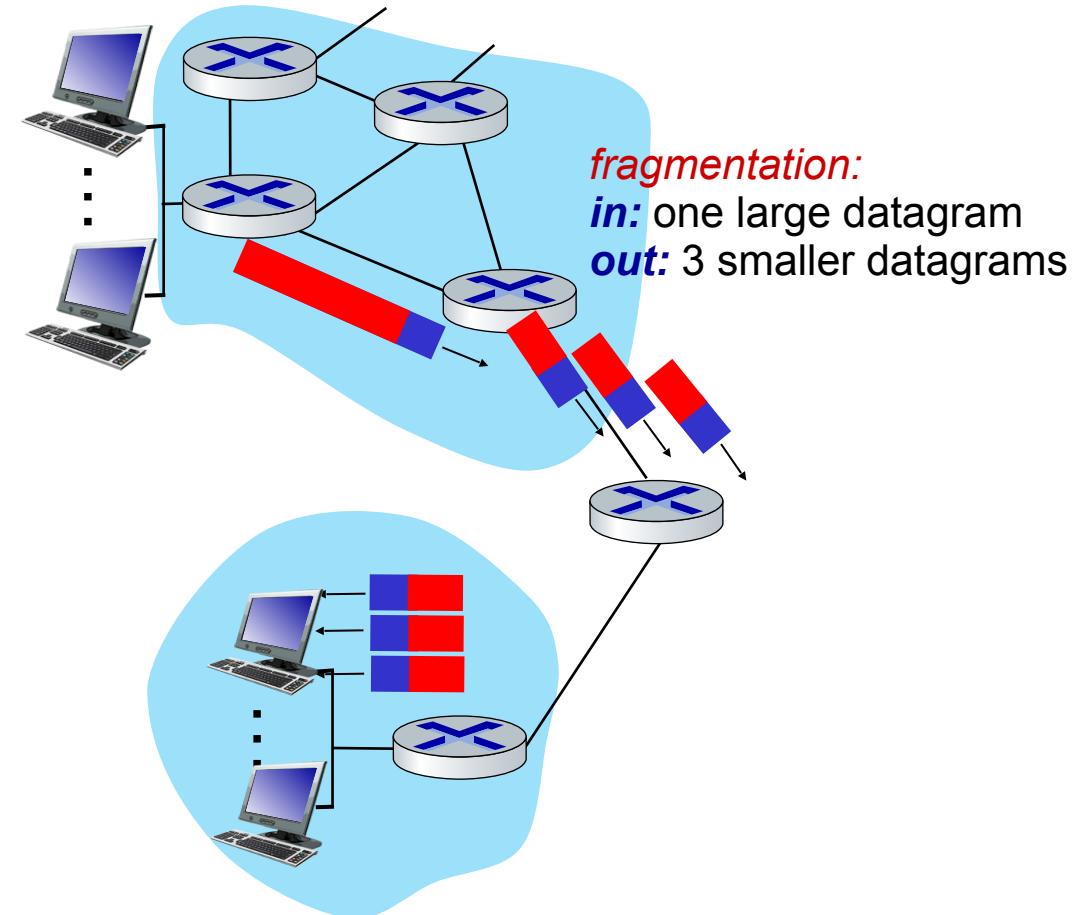
IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



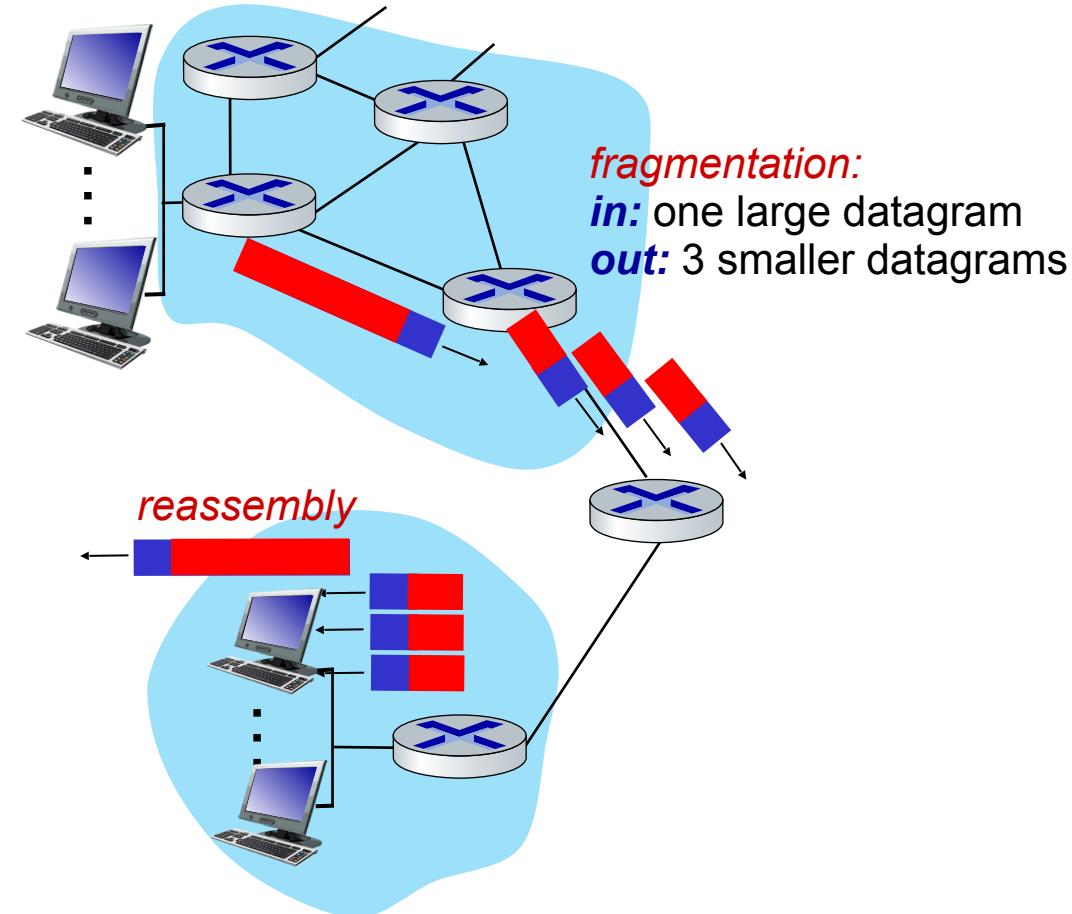
IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



IP fragmentation/reassembly

- network links have MTU (max. transfer size) - largest possible link-level frame
 - different link types, different MTUs
- large IP datagram divided (“fragmented”) within net
 - one datagram becomes several datagrams
 - “reassembled” only at *destination*
 - IP header bits used to identify, order related fragments



IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

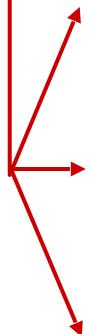
	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

one large datagram becomes several smaller datagrams

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--



IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

1480 bytes in
data field

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

*one large datagram becomes
several smaller datagrams*

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

IP fragmentation/reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes

1480 bytes in
data field

offset =
 $1480/8$

	length =4000	ID =x	fragflag =0	offset =0	
--	-----------------	----------	----------------	--------------	--

*one large datagram becomes
several smaller datagrams*

	length =1500	ID =x	fragflag =1	offset =0	
--	-----------------	----------	----------------	--------------	--

	length =1500	ID =x	fragflag =1	offset =185	
--	-----------------	----------	----------------	----------------	--

	length =1040	ID =x	fragflag =0	offset =370	
--	-----------------	----------	----------------	----------------	--

Chapter 4: roadmap

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

ICMP: internet control message protocol

ICMP: internet control message protocol

- used by hosts and routers to communicate network-level information
 - error reporting: unreachable host, network, port, protocol
 - echo request/reply (used by ping)

ICMP: internet control message protocol

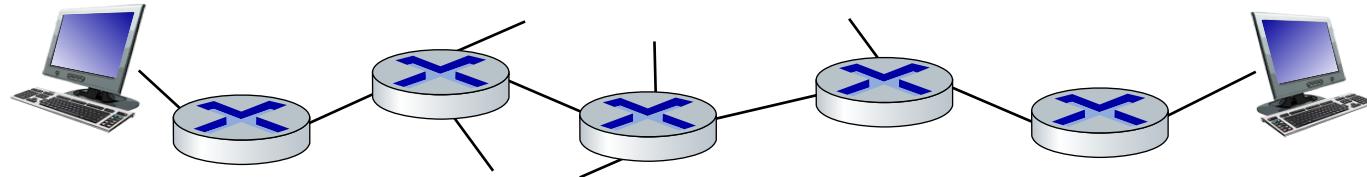
- used by hosts and routers to communicate network-level information
 - error reporting: unreachable host, network, port, protocol
 - echo request/reply (used by ping)
- network-layer “above” IP:
 - ICMP messages carried in IP datagrams

ICMP: internet control message protocol

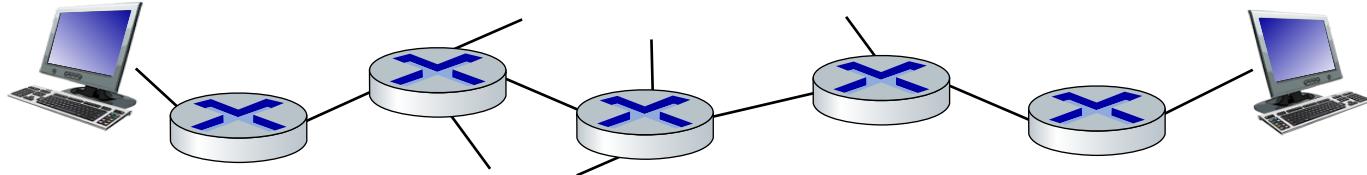
- used by hosts and routers to communicate network-level information
 - error reporting: unreachable host, network, port, protocol
 - echo request/reply (used by ping)
- network-layer “above” IP:
 - ICMP messages carried in IP datagrams
- *ICMP message:* type, code plus first 8 bytes of IP datagram causing error

Type	Code	description
0	0	echo reply (ping)
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

Traceroute and ICMP

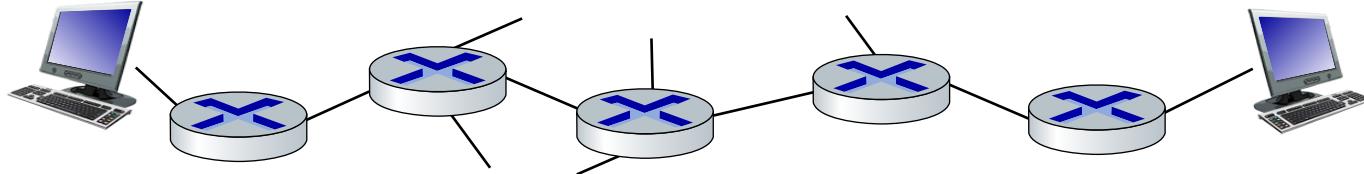


Traceroute and ICMP



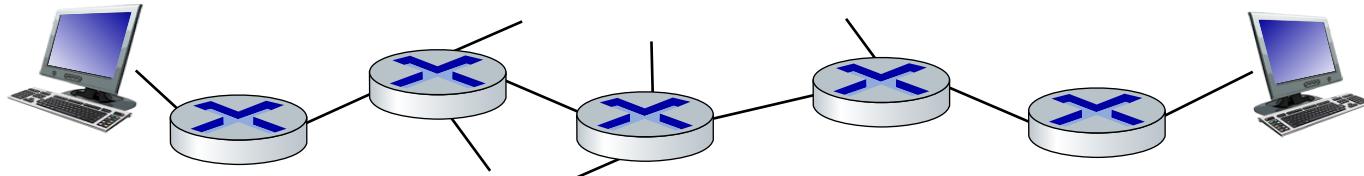
- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.

Traceroute and ICMP



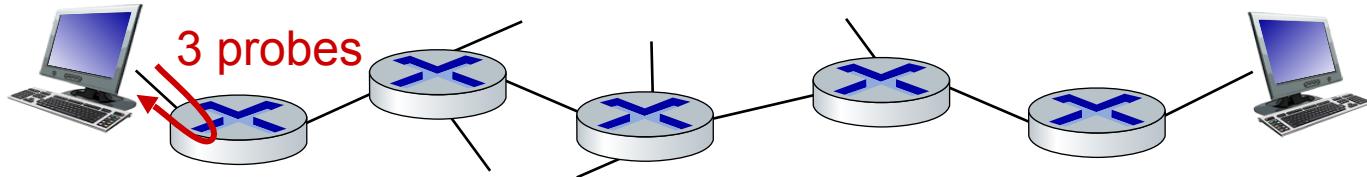
- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in *n*th set arrives to *n*th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address

Traceroute and ICMP



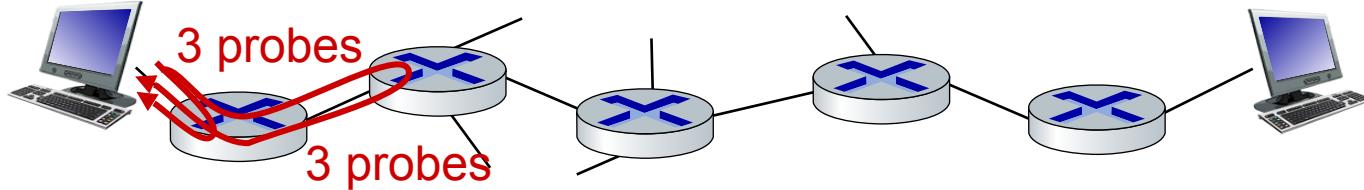
- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

Traceroute and ICMP



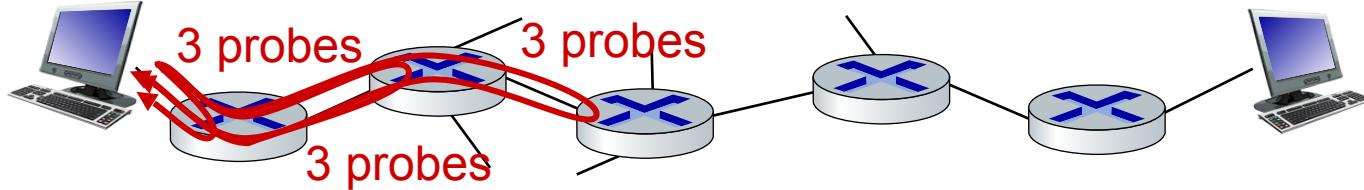
- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

Traceroute and ICMP



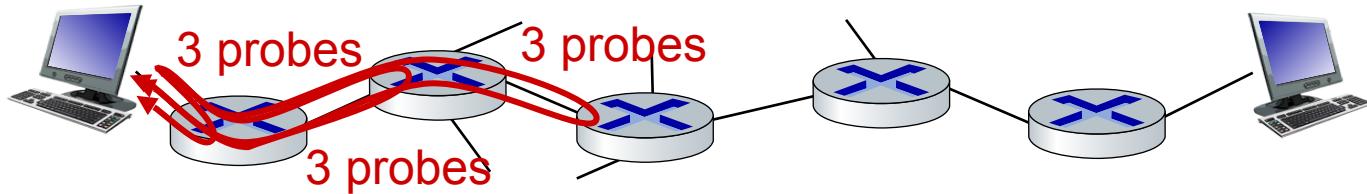
- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

Traceroute and ICMP



- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

Traceroute and ICMP

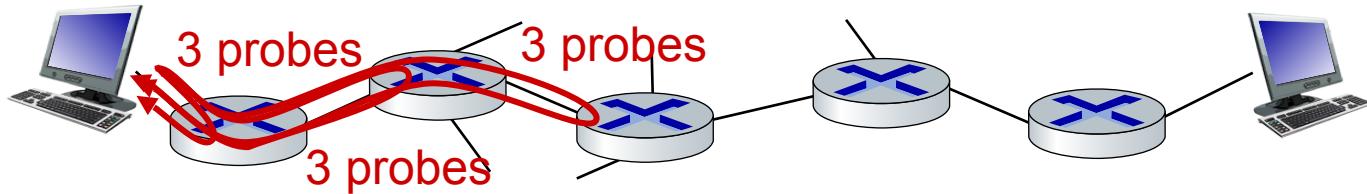


- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” message (type 3, code 3)
- source stops

Traceroute and ICMP



- source sends sets of UDP segments to destination
 - 1st set has TTL =1, 2nd set has TTL=2, etc.
- datagram in n th set arrives to n th router:
 - router discards datagram and sends source ICMP message (type 11, code 0)
 - ICMP message possibly includes name of router & IP address
- when ICMP message arrives at source: record RTTs

stopping criteria:

- UDP segment eventually arrives at destination host
- destination returns ICMP “port unreachable” message (type 3, code 3)
- source stops

Chapter 4: roadmap

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

IPv6: motivation

IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated

IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated
- additional motivation:
 - speed processing/forwarding: 40-byte fixed length header

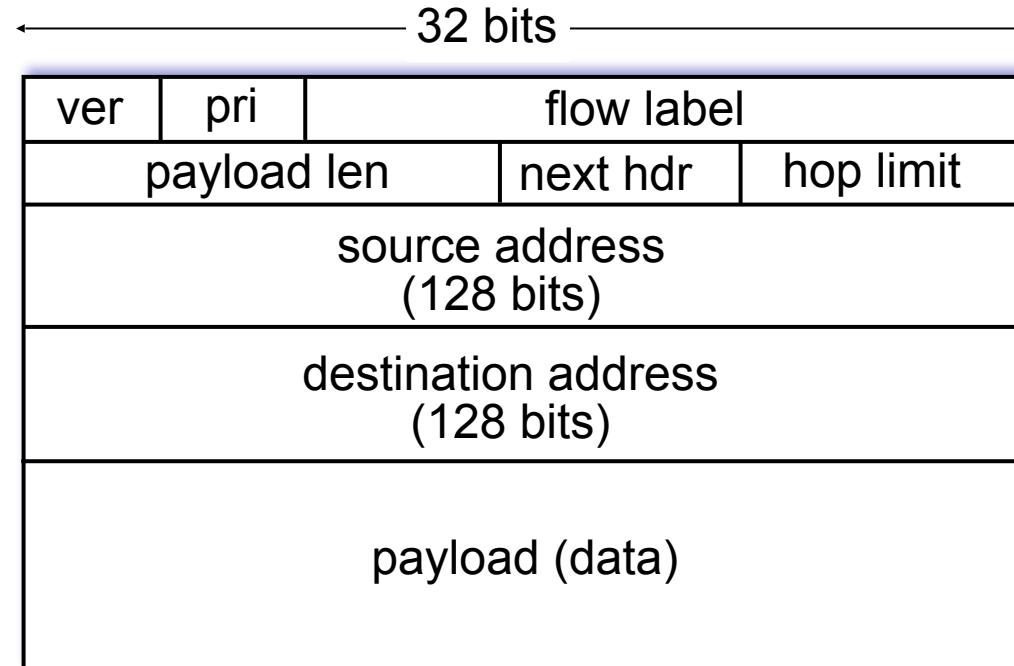
IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated
- additional motivation:
 - speed processing/forwarding: 40-byte fixed length header
 - enable different network-layer treatment of “flows”

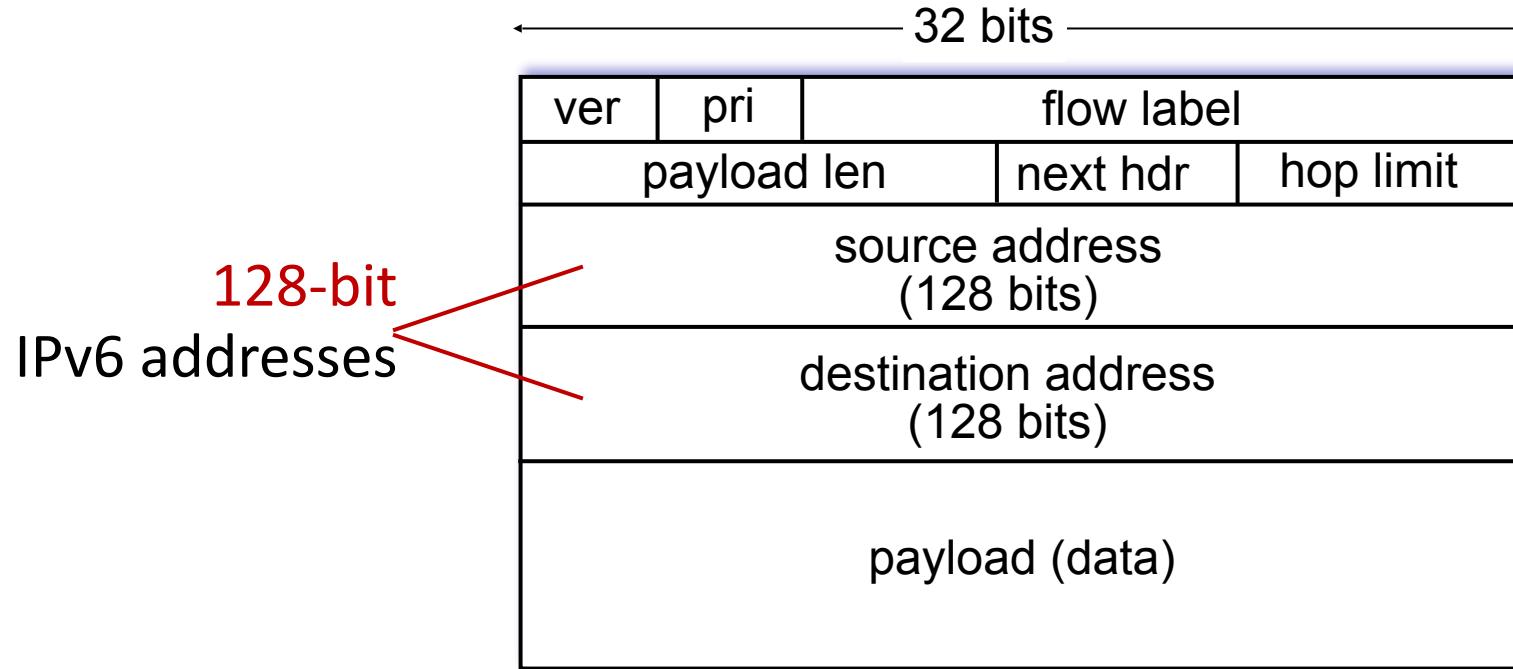
IPv6: motivation

- **initial motivation:** 32-bit IPv4 address space would be completely allocated
- additional motivation:
 - speed processing/forwarding: 40-byte fixed length header
 - enable different network-layer treatment of “flows”

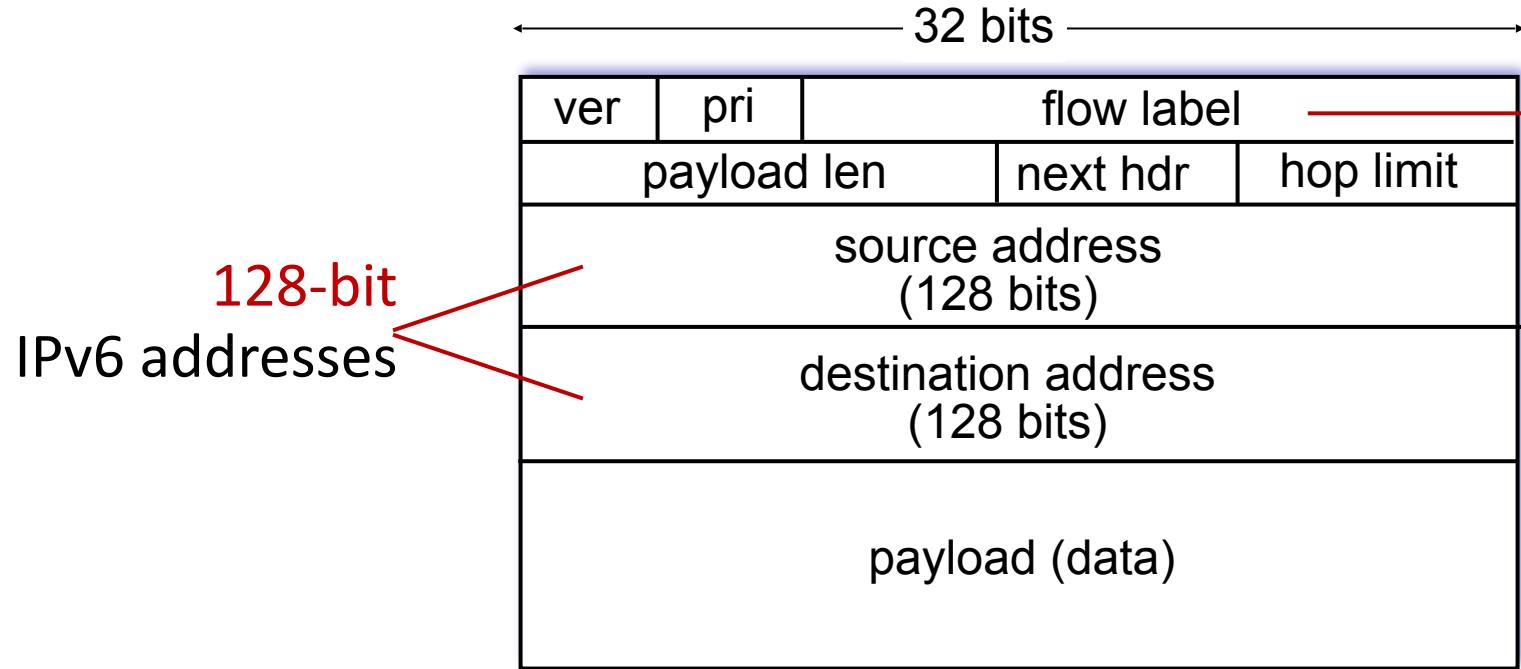
IPv6 datagram format



IPv6 datagram format



IPv6 datagram format

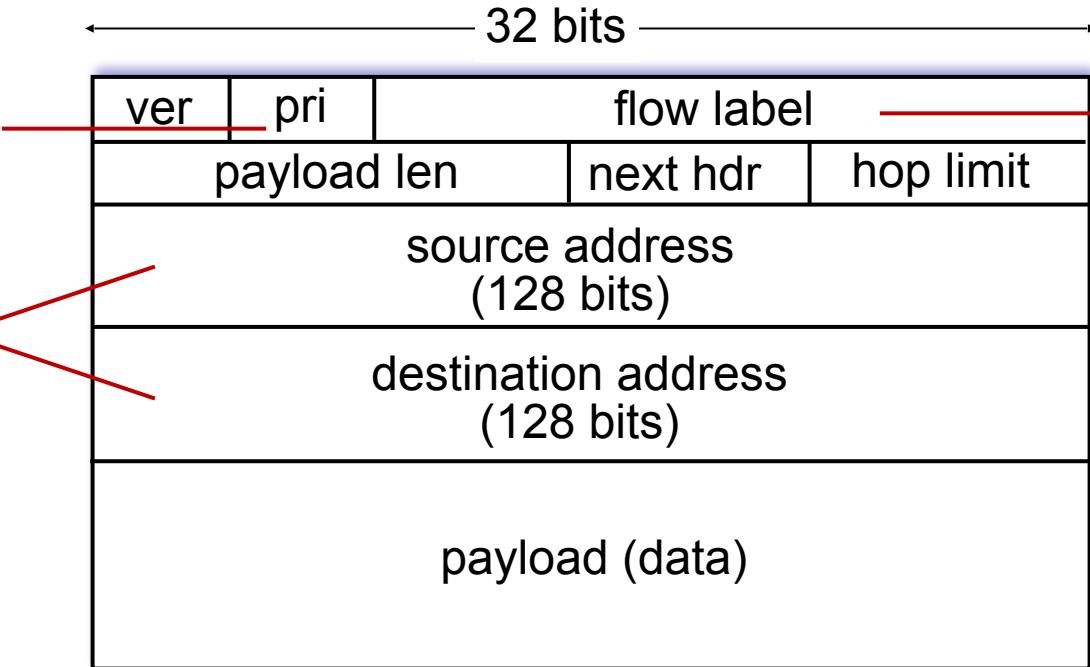


flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

IPv6 datagram format

priority: identify priority among datagrams in flow

128-bit IPv6 addresses

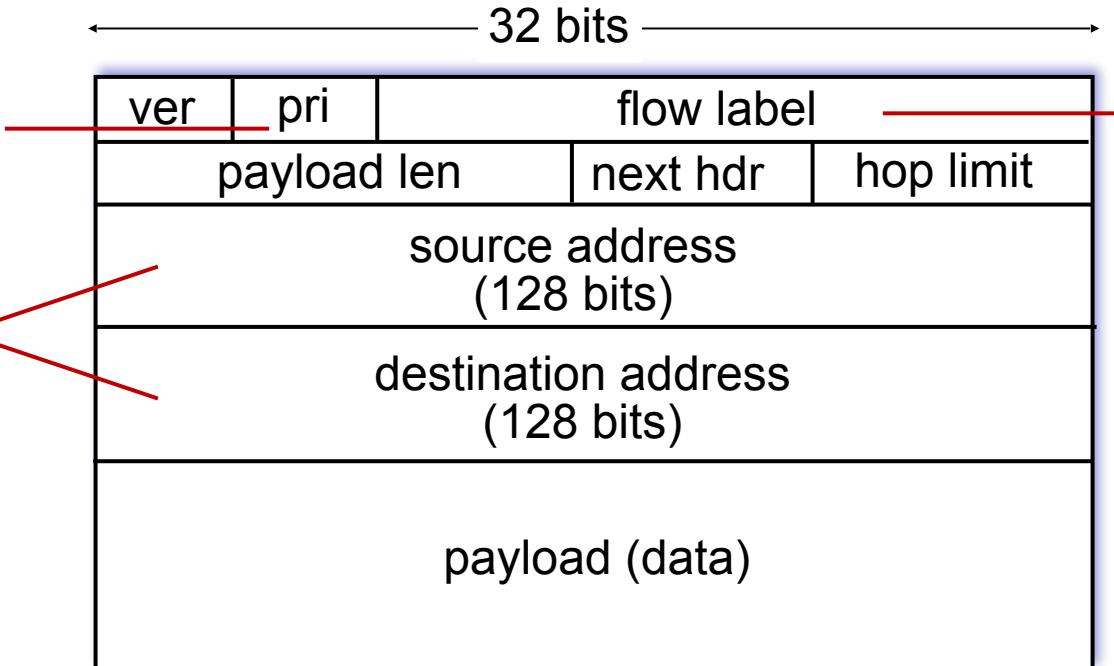


flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

IPv6 datagram format

priority: identify priority among datagrams in flow

**128-bit
IPv6 addresses**



flow label: identify datagrams in same "flow." (concept of "flow" not well defined).

What's missing (compared with IPv4):

- no checksum (to speed processing at routers)
- no fragmentation/reassembly
- no options (available as upper-layer, next-header protocol at router)

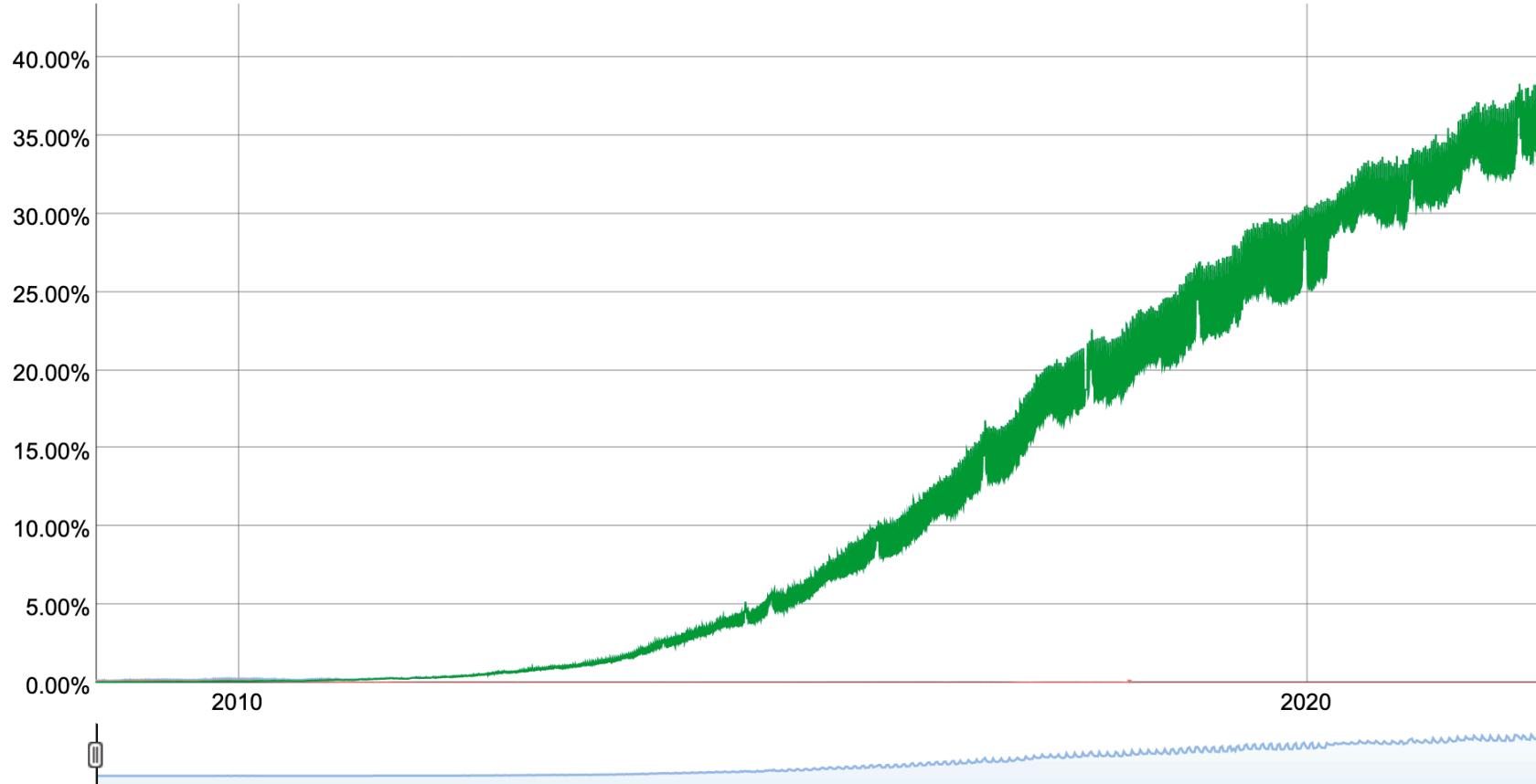
IPv6: adoption

- Google¹: ~ 38% of clients access services via IPv6

IPv6 Adoption

We are continuously measuring the availability of IPv6 connectivity among Google users. The graph shows the percentage of users that access Google over IPv6.

Native: 38.04% 6to4/Teredo: 0.00% Total IPv6: 38.04% | Apr 3, 2022



IPv6: adoption

- Google¹: ~ 38% of clients access services via IPv6
 - Portugal: 31% of all requests
- DNS.pt: 45% of DNS requests occurred over IPv6 (Dec. 2020)
- Long (long!) time for deployment, use
 - 25 years and counting!
 - think of application-level changes in last 25 years: WWW, social media, streaming media, gaming, telepresence, ...
 - *Why?*

¹ <https://www.google.com/intl/en/ipv6/statistics.html>

Chapter 4: done!

- Network layer: overview
 - data plane
 - control plane
- What's inside a router
 - input ports, switching, output ports
 - buffer management, scheduling
- IP: the Internet Protocol
 - datagram format
 - IPv4 addressing
 - network address translation
 - fragmentation
 - ICMP
 - IPv6

Additional Chapter 4 slides

Sidebar: Network Neutrality

What is network neutrality?

- *technical*: how an ISP should share/allocation its resources
 - packet scheduling, buffer management are the *mechanisms*
- *social, economic* principles
 - protecting free speech
 - encouraging innovation, competition
- enforced *legal* rules and policies

Different countries have different “takes” on network neutrality

Sidebar: Network Neutrality

2015 US FCC *Order on Protecting and Promoting an Open Internet*: three “clear, bright line” rules:

- **no blocking** ... “shall not block lawful content, applications, services, or non-harmful devices, subject to reasonable network management.”
- **no throttling** ... “shall not impair or degrade lawful Internet traffic on the basis of Internet content, application, or service, or use of a non-harmful device, subject to reasonable network management.”
- **no paid prioritization.** ... “shall not engage in paid prioritization”

ISP: telecommunications or information service?

Is an ISP a “telecommunications service” or an “information service” provider?

- the answer *really* matters from a regulatory standpoint!

US Telecommunication Act of 1934 and 1996:

- *Title II*: imposes “common carrier duties” on *telecommunications services*: reasonable rates, non-discrimination and *requires regulation*
- *Title I*: applies to *information services*:
 - no common carrier duties (*not regulated*)
 - but grants FCC authority “... as may be necessary in the execution of its functions”

DHCP: Wireshark output (home LAN)

Message type: **Boot Request (1)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 0.0.0.0 (0.0.0.0)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 0.0.0.0 (0.0.0.0)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) **DHCP Message Type = DHCP Request**

Option: (61) Client identifier

Length: 7; Value: 010016D323688A;

Hardware type: Ethernet

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Option: (t=50,l=4) Requested IP Address = 192.168.1.101

Option: (t=12,l=5) Host Name = "nomad"

Option: (55) Parameter Request List

Length: 11; Value: 010F03062C2E2F1F21F92B

1 = Subnet Mask; 15 = Domain Name

3 = Router; 6 = Domain Name Server

44 = NetBIOS over TCP/IP Name Server

.....

request

Message type: **Boot Reply (2)**

Hardware type: Ethernet

Hardware address length: 6

Hops: 0

Transaction ID: 0x6b3a11b7

Seconds elapsed: 0

Bootp flags: 0x0000 (Unicast)

Client IP address: 192.168.1.101 (192.168.1.101)

Your (client) IP address: 0.0.0.0 (0.0.0.0)

Next server IP address: 192.168.1.1 (192.168.1.1)

Relay agent IP address: 0.0.0.0 (0.0.0.0)

Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)

Server host name not given

Boot file name not given

Magic cookie: (OK)

Option: (t=53,l=1) DHCP Message Type = DHCP ACK

Option: (t=54,l=4) Server Identifier = 192.168.1.1

Option: (t=1,l=4) Subnet Mask = 255.255.255.0

Option: (t=3,l=4) Router = 192.168.1.1

Option: (6) Domain Name Server

Length: 12; Value: 445747E2445749F244574092;

IP Address: 68.87.71.226;

IP Address: 68.87.73.242;

IP Address: 68.87.64.146

Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

reply